

Panduan Pengembang untuk versi 1.x

AWS SDK for Java 1.x



AWS SDK for Java 1.x: Panduan Pengembang untuk versi 1.x

Table of Contents

.....	viii
AWS SDK for Java1.x	1
Versi 2 SDK dirilis	1
Dokumentasi dan Sumber Daya Tambahan	1
Eclipse Support	2
Mengembangkan Aplikasi untuk Android	2
Melihat Riwayat Revisi SDK	2
Membangun Dokumentasi Referensi Java untuk versi SDK Sebelumnya	2
Memulai	4
Pengaturan dasar	4
Gambaran Umum	4
Kemampuan masuk ke portal akses AWS	5
Menyiapkan file konfigurasi bersama	5
Instal Lingkungan Pengembangan Java	7
Cara untuk mendapatkan AWS SDK for Java	8
Prasyarat	8
Gunakan alat build	8
Unduh toples bawaan	8
Membangun dari sumber	9
Ketahuilah alat bangun	10
Menggunakan SDK dengan Apache Maven	10
Menggunakan SDK dengan Gradle	13
Kredensial sementara dan Wilayah	17
Konfigurasi kredensial sementara	17
Kredensial IMDS yang menyegarkan	18
Mengatur Wilayah AWS	19
Menggunakan AWS SDK for Java	21
Praktik Terbaik untuk AWS Pengembangan dengan AWS SDK for Java	21
S3	21
Membuat Klien Layanan	22
Memperoleh Client Builder	23
Membuat Klien Async	24
Menggunakan DefaultClient	24
Siklus Hidup Klien	25

Berikan kredensial sementara	25
Menggunakan Rantai Penyedia Kredensial Default	26
Tentukan penyedia kredensial atau rantai penyedia	29
Secara eksplisit menentukan kredensial sementara	30
Info Selengkapnya	30
Wilayah AWS Seleksi	31
Memeriksa Ketersediaan Layanan di Wilayah	31
Memilih Wilayah	31
Memilih Endpoint Tertentu	32
Secara Otomatis Menentukan Wilayah dari Lingkungan	32
Penanganan Pengecualian	34
Mengapa Pengecualian Tidak Dicentang?	34
AmazonServiceException (dan Subclass)	35
AmazonClientException	35
Pemrograman Asinkron	35
Java Berjangka	36
Callback Asinkron	37
Praktik Terbaik	39
AWS SDK for Java Panggilan Pencatatan	40
Unduh Log4J JAR	40
Mengatur Classpath	41
Kesalahan dan Peringatan Khusus Layanan	41
Pencatatan Ringkasan Permintaan/Tanggapan	42
Penebangan Kawat Verbose	43
Pencatatan Metrik Latensi	43
Konfigurasi Klien	44
Konfigurasi Proxy	44
Konfigurasi Transportasi HTTP	44
Petunjuk Ukuran Penyangga Soket TCP	46
Kebijakan Kontrol Akses	47
Amazon S3 Contoh	47
Amazon SQS Contoh	48
Contoh Amazon SNS	48
Mengatur JVM TTL untuk pencarian nama DNS	49
Cara mengatur JVM TTL	49
Mengaktifkan Metrik untuk AWS SDK for Java	50

Cara Mengaktifkan Generasi Metrik SDK Java	50
Jenis Metrik yang Tersedia	51
Informasi Selengkapnya	54
Contoh Kode	56
AWS SDK for Java2.x	56
Contoh Amazon CloudWatch	56
Mendapatkan Metrik dari CloudWatch	57
Memublikasikan Data Metrik Khusus	59
Bekerja dengan CloudWatch Alarm	60
Menggunakan Tindakan Alarm di CloudWatch	63
Mengirim Peristiwa keCloudWatch	65
Contoh Amazon DynamoDB	68
Cara dengan denganDynamoDB	68
Cara Menggunakan ItemDynamoDB	75
Contoh Amazon EC2	82
Tutorial: Memulai instans EC2	82
Menggunakan Peran IAM untuk Memberikan Akses keAWSSumber Daya diAmazon EC2	88
Tutorial:Amazon EC2 Instans Spot	94
Tutorial: LanjutanAmazon EC2Manajemen permintaan spot	106
MengelolaAmazon EC2Instans	123
Menggunakan Alamat IP Elastis di Amazon EC2	128
Gunakan wilayah dan availability zone	131
Bekerja denganAmazon EC2Pasangan Kunci	134
Bekerja dengan GrupAmazon EC2	136
AWS Identity and Access Management(IAM) Contoh	140
Mengelola Kunci Akses IAM	141
Membuat pengguna IAM	145
Menggunakan Alias Akun IAM	149
Cara menggunakan kebijakan IAM	151
Cara menggunakan Sertifikat Server IAM	156
AmazonLambdaContoh	160
Operasi Layanan	160
Contoh Amazon Pinpoint	164
Membuat dan Menghapus AplikasiAmazon Pinpoint	164
Membuat Endpoint diAmazon Pinpoint	166
Membuat Segmen diAmazon Pinpoint	168

Membuat Kampanye diAmazon Pinpoint	170
Saluran diAmazon Pinpoint	171
Contoh Amazon S3	173
Membuat, Daftar, dan MenghapusAmazon S3Bucket	173
Melakukan Operasi pada Amazon S3 Objek	178
MengelolaAmazon S3Akses Izin untuk Bucket dan Objek	183
Mengelola akses keAmazon S3Bucket Menggunakan Kebijakan ember	187
Menggunakan TransferManager untukAmazon S3Operasi	191
MengonfigurasiAmazon S3Bucket sebagai Situs Web	203
GunakanAmazon S3Enkripsi sisi klien	207
Contoh Amazon SQS	213
Bekerja denganAmazon SQSAntrean pesan	214
Mengirim, Menerima, dan MenghapusAmazon SQSPesan	217
Mengaktifkan Jajak pendapat panjangAmazon SQSAntrean	219
Mengatur Timeout Visibilitas diAmazon SQS	221
Menggunakan Antrian Surat Mati diAmazon SQS	224
Contoh Amazon SWF	226
Dasar-dasar SWF	227
Membangun Amazon SWF Aplikasi Sederhana	229
Tugas Lambda	248
Mematikan Kegiatan dan Alur Kerja Pekerja Anggun	253
Mendaftarkan domain	256
Daftar domain	257
Contoh Kode disertakan dengan SDK	257
Cara Mendapatkan Sampel	258
Membangun dan Menjalankan Sampel Menggunakan Command Line	258
Membangun dan Menjalankan Sampel Menggunakan Eclipse IDE	259
Keamanan	261
Perlindungan data	261
Menegakkan versi TLS minimum	262
Cara memeriksa versi TLS	263
Menegakkan versi TLS minimum	263
Identity and Access Management	263
Audiens	264
Mengautentikasi dengan identitas	264
Mengelola akses menggunakan kebijakan	268

Bagaimana Layanan AWS bekerja dengan IAM	271
Memecahkan masalah AWS identitas dan akses	271
Validasi Kepatuhan	273
Ketangguhan	275
Keamanan Infrastruktur	275
Migrasi Klien Enkripsi S3	276
Prasyarat	276
Ikhtisar Migrasi	276
Perbarui Klien yang Ada untuk Membaca Format Baru	277
Migrasi Klien Enkripsi dan Dekripsi ke V2	278
Contoh Tambahan	280
Kunci OpenPGP	282
Kunci saat ini	282
Riwayat Dokumen	284

Kami [mengumumkan](#) yang akan datang end-of-support untuk AWS SDK for Java (v1). Kami menyarankan Anda bermigrasi ke [AWS SDK for Java v2](#). Untuk tanggal, detail tambahan, dan informasi tentang cara bermigrasi, silakan merujuk ke pengumuman tertaut.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.

Panduan Pengembang -AWS SDK for Java 1.x

[AWS SDK for Java](#) menyediakan API Java untuk AWS layanan. Menggunakan SDK, Anda dapat dengan mudah membangun aplikasi Java yang bekerja dengan Amazon S3, Amazon EC2, DynamoDB, dan banyak lagi. Kami secara teratur menambahkan dukungan untuk layanan baru ke AWS SDK for Java. Untuk daftar layanan yang didukung dan versi API mereka yang disertakan dengan setiap rilis SDK, lihat [catatan rilis](#) untuk versi yang sedang Anda kerjakan.

Versi 2 SDK dirilis

Lihatlah AWS SDK for Java 2.x baru di <https://github.com/aws/aws-sdk-java-v2/>. Ini mencakup banyak fitur yang ditunggu, seperti cara untuk mencolokkan implementasi HTTP. Untuk memulai, lihat [AWS SDK for Java 2.x Developer Guide](#).

Dokumentasi dan Sumber Daya Tambahan

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

- [AWS SDK for Java Referensi API](#)
- [Blog pengembang Java](#)
- [Forum pengembang Java](#)
- GitHub:
 - [Sumber dokumentasi](#)
 - [Masalah dokumentasi](#)
 - [Sumber SDK](#)
 - [Masalah SDK](#)
 - [Sampel SDK](#)
 - [Saluran Gitter](#)
- Yang [Katalog Kode Sampel AWS](#)
- [@awsforjava](#) (Kerica)
- [catatan rilis](#)

Eclipse Support

Jika Anda mengembangkan kode menggunakan Eclipse IDE, Anda dapat menggunakan [AWS Toolkit for Eclipse](#) untuk menambahkan AWS SDK for Java ke proyek Eclipse yang ada atau untuk membuat AWS SDK for Java proyek baru. Toolkit ini juga mendukung pembuatan dan pengunggahan Lambda fungsi, peluncuran dan pemantauan Amazon EC2 instans, mengelola IAM pengguna dan grup keamanan, editor AWS CloudFormation template, dan banyak lagi.

Lihat [Panduan AWS Toolkit for Eclipse Pengguna](#) untuk dokumentasi lengkap.

Mengembangkan Aplikasi untuk Android

Jika Anda pengembang Android, Amazon Web Services publikasikan SDK yang dibuat khusus untuk pengembangan Android: [Amplify Android \(AWS Mobile SDK for Android\)](#).

Melihat Riwayat Revisi SDK

Untuk melihat riwayat rilis AWS SDK for Java, termasuk perubahan dan layanan yang didukung per versi SDK, lihat [catatan rilis](#) SDK.

Membangun Dokumentasi Referensi Java untuk versi SDK Sebelumnya

[Referensi AWS SDK for Java API](#) mewakili build terbaru versi 1.x SDK. Jika Anda menggunakan versi 1.x versi sebelumnya, Anda mungkin ingin mengakses dokumentasi referensi SDK yang cocok dengan versi yang Anda gunakan.

Cara termudah untuk membangun dokumentasi menggunakan Apache [Maven](#) membangun alat. Download dan instal Maven pertama jika Anda belum memilikinya pada sistem Anda, kemudian gunakan petunjuk berikut untuk membangun dokumentasi referensi.

1. Temukan dan pilih versi SDK yang Anda gunakan di halaman [rilis](#) repositori SDK GitHub.
2. Pilih tautan `zip` (sebagian besar platform, termasuk Windows) atau `tar.gz` (Linux, macOS, atau Unix) untuk mengunduh SDK ke komputer Anda.
3. Buka arsip ke direktori lokal.
4. Pada baris perintah, navigasikan ke direktori tempat Anda membongkar arsip, dan ketik yang berikut.

```
mvn javadoc:javadoc
```

5. Setelah membangun selesai, Anda akan menemukan dokumentasi HTML yang dihasilkan dalam `aws-java-sdk/target/site/apidocs/` direktori.

Memulai

Bagian ini menyediakan informasi tentang cara menginstal, menyiapkan, dan menggunakan AWS SDK for Java.

Topik

- [Pengaturan dasar untuk bekerja dengan Layanan AWS](#)
- [Cara untuk mendapatkan AWS SDK for Java](#)
- [Ketahuilah alat bangun](#)
- [Menyiapkan kredensial AWS sementara dan Wilayah AWS untuk pengembangan](#)

Pengaturan dasar untuk bekerja dengan Layanan AWS

Gambaran Umum

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 diperlukan aplikasi Anda. Izin yang terkait dengan kebijakan yang `PowerUserAccessAWS` dikelola cukup untuk sebagian besar kebutuhan pengembangan.
- Lingkungan pengembangan dengan elemen berikut:
 - [File konfigurasi bersama](#) yang disiapkan dengan cara berikut:
 - `configFile` berisi profil default yang menentukan file. Wilayah AWS
 - `credentialsFile` berisi kredensial sementara sebagai bagian dari profil default.
 - [Instalasi Java yang](#) cocok.
 - [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](#)

Bila Anda menggunakan IDE, Anda juga dapat mengintegrasikan AWS Toolkit s untuk lebih mudah bekerja dengan Layanan AWS. [AWS Toolkit for IntelliJ](#) Dan [AWS Toolkit for Eclipse](#) dua toolkit yang dapat Anda gunakan untuk pengembangan Java.

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

Kemampuan masuk ke portal akses AWS

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

Jika Anda tidak terbiasa dengan portal AWS akses, ikuti panduan untuk akses akun di [Langkah 1 topik otentikasi IAM Identity Center](#) di Panduan Referensi AWS SDK dan Alat. Jangan ikuti Langkah 2 karena AWS SDK for Java 1.x tidak mendukung penyegaran token otomatis dan pengambilan otomatis kredensial sementara untuk SDK yang dijelaskan oleh Langkah 2.

Menyiapkan file konfigurasi bersama

File konfigurasi bersama berada di workstation pengembangan Anda dan berisi pengaturan dasar yang digunakan oleh semua AWS SDK dan AWS Command Line Interface (CLI). File konfigurasi bersama dapat berisi [sejumlah pengaturan](#), tetapi instruksi ini mengatur elemen dasar yang diperlukan untuk bekerja dengan SDK.

Atur **config** file bersama

Contoh berikut menunjukkan isi dari `config` file bersama.

```
[default]
region=us-east-1
output=json
```

Untuk tujuan pengembangan, gunakan yang Wilayah AWS [terdekat](#) ke tempat Anda berencana untuk menjalankan kode Anda. Untuk [daftar kode wilayah](#) yang akan digunakan dalam config file, lihat Referensi Umum Amazon Web panduan. jsonPengaturan untuk format output adalah salah satu dari [beberapa nilai yang mungkin](#).

Ikuti panduan [di bagian ini](#) untuk membuat config file.

Menyiapkan kredensi sementara untuk SDK

Setelah Anda memiliki akses ke peran Akun AWS dan IAM melalui portal AWS akses, konfigurasi lingkungan pengembangan Anda dengan kredensi sementara agar SDK dapat diakses.

Langkah-langkah untuk menyiapkan **credentials** file lokal dengan kredensi sementara

1. [Buat credentials file bersama](#).
2. Dalam credentials file, tempelkan teks placeholder berikut sampai 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>
```

3. Simpan file tersebut. File sekarang `~/.aws/credentials` harus ada pada sistem pengembangan lokal Anda. File ini berisi [profil \[default\]](#) yang digunakan SDK for Java jika profil bernama tertentu tidak ditentukan.
4. [Masuk ke portal AWS akses](#).
5. Ikuti petunjuk ini di bawah judul [Penyegaran kredensi manual](#) untuk menyalin kredensi peran IAM dari portal akses. AWS
 - a. Untuk langkah 4 dalam petunjuk tertaut, pilih nama peran IAM yang memberikan akses untuk kebutuhan pengembangan Anda. Peran ini biasanya memiliki nama seperti PowerUserAccessatau Pengembang.
 - b. Untuk langkah 7, pilih Tambahkan profil secara manual ke file AWS kredensial Anda opsi dan salin isinya.

Cara untuk mendapatkan AWS SDK for Java

Prasyarat

Untuk menggunakan AWS SDK for Java, Anda harus memiliki:

- Anda harus dapat [masuk ke portal AWS akses](#) yang tersedia di AWS IAM Identity Center.
- [Instalasi Java yang](#) cocok.
- Kredensial sementara disiapkan di file bersama `credentials` lokal Anda.

Lihat [the section called “Pengaturan dasar”](#) topik untuk petunjuk tentang cara menyiapkan untuk menggunakan SDK for Java.

Gunakan alat build untuk mengelola dependensi SDK for Java (disarankan)

Sebaiknya gunakan Apache Maven atau Gradle dengan project Anda untuk mengakses dependensi SDK for Java yang diperlukan. [Bagian ini](#) menjelaskan cara menggunakan alat-alat tersebut.

Unduh dan ekstrak SDK (tidak disarankan)

Kami menyarankan Anda menggunakan alat build untuk mengakses SDK untuk proyek Anda. Namun, Anda dapat mengunduh toples bawaan SDK versi terbaru.

Note

Untuk selengkapnya tentang cara mengunduh dan membuat versi SDK sebelumnya, lihat [Menginstal SDK versi sebelumnya](#).

1. Unduh SDK dari <https://sdk-for-java.amazonwebservices.com/latest/.zip.aws-java-sdk>
2. Setelah mengunduh SDK, ekstrak konten ke direktori lokal.

SDK berisi direktori berikut:

- `documentation-` berisi dokumentasi API (juga tersedia di web: [Referensi AWS SDK for Java API](#)).

- `lib-` berisi `.jar` file SDK.
- `samples-` berisi kode contoh kerja yang menunjukkan cara menggunakan SDK.
- `third-party/lib-` berisi pustaka pihak ketiga yang digunakan oleh SDK, seperti Apache commons logging, aspectJ dan framework Spring.

Untuk menggunakan SDK, tambahkan path lengkap ke `third-party` direktori `lib` dan ke dependensi dalam file build Anda, dan tambahkan ke java CLASSPATH Anda untuk menjalankan kode Anda.

Buat versi SDK sebelumnya dari sumber (tidak disarankan)

Hanya versi terbaru dari SDK lengkap yang disediakan dalam bentuk pra-bangun sebagai toples yang dapat diunduh. Namun, Anda dapat membangun versi SDK sebelumnya menggunakan Apache Maven (open source). Maven akan mengunduh semua dependensi yang diperlukan, membangun dan menginstal SDK dalam satu langkah. Kunjungi <http://maven.apache.org/> untuk petunjuk instalasi dan informasi lebih lanjut.

1. Buka GitHub halaman SDK di: [AWS SDK for Java \(GitHub\)](#).
2. Pilih tag yang sesuai dengan nomor versi SDK yang Anda inginkan. Misalnya, `1.6.10`.
3. Klik tombol Unduh ZIP untuk mengunduh versi SDK yang Anda pilih.
4. Buka zip file ke direktori di sistem pengembangan Anda. Pada banyak sistem, Anda dapat menggunakan manajer file grafis Anda untuk melakukan ini, atau menggunakan `unzip` utilitas di jendela terminal.
5. Di jendela terminal, arahkan ke direktori tempat Anda membuka ritsleting sumber SDK.
6. Bangun dan instal SDK dengan perintah berikut (diperlukan [Maven](#)):

```
mvn clean install -Dpgp.skip=true
```

`.jarFile` yang dihasilkan dibangun ke dalam `target` direktori.

7. (Opsional) Buat dokumentasi Referensi API menggunakan perintah berikut:

```
mvn javadoc:javadoc
```

Dokumentasi dibangun ke dalam `target/site/apidocs/` direktori.

Ketahui alat bangun

Penggunaan alat build membantu mengelola pengembangan proyek Java. Beberapa alat build tersedia, tetapi kami menunjukkan cara bangun dan berjalan dengan dua alat build populer-- Maven dan Gradle. Topik ini menunjukkan kepada Anda cara menggunakan alat build ini mengelola dependensi SDK for Java yang Anda butuhkan untuk proyek Anda.

Topik

- [Menggunakan SDK dengan Apache Maven](#)
- [Menggunakan SDK dengan Gradle](#)

Menggunakan SDK dengan Apache Maven

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

Note

Anda harus memiliki Maven diinstal untuk menggunakan panduan dalam topik ini. Jika belum diinstal, kunjungi <http://maven.apache.org/> untuk mengunduh dan menginstalnya.

Membuat paket Maven baru

Untuk membuat paket Maven dasar, buka jendela terminal (command-line) dan jalankan:

```
mvn -B archetype:generate \  
  -DarchetypeGroupId=org.apache.maven.archetypes \  
  -DgroupId=org.example.basicapp \  
  -DartifactId=myapp
```

Ganti org.example.basicapp dengan namespace paket lengkap aplikasi Anda, dan myapp dengan nama proyek Anda (ini akan menjadi nama direktori untuk proyek Anda).

Secara default, membuat template proyek untuk Anda menggunakan pola dasar [mulai cepat](#), yang merupakan tempat awal yang baik untuk banyak proyek. Ada lebih banyak arketipe yang tersedia; kunjungi halaman [arketipe Maven](#) untuk daftar arketipe yang dikemas. Anda dapat memilih pola

dasar tertentu untuk digunakan dengan menambahkan `-DarchetypeArtifactId` argumen `kearchetype:generate` perintah. Misalnya:

```
mvn archetype:generate \  
-DarchetypeGroupId=org.apache.maven.archetypes \  
-DarchetypeArtifactId=maven-archetype-webapp \  
-DgroupId=org.example.webapp \  
-DartifactId=mywebapp
```

Note

Lebih banyak informasi tentang membuat dan mengkonfigurasi proyek disediakan dalam [Panduan Memulai Maven](#).

Mengkonfigurasi SDK sebagai dependensi Maven

Untuk menggunakan AWS SDK for Java dalam proyek Anda, Anda harus mendeklarasikannya sebagai dependensi dalam `pom.xml` file proyek Anda. Dimulai dengan versi 1.9.0, Anda dapat mengimpor [komponen individual](#) atau [seluruh SDK](#).

Menentukan modul SDK individu

Untuk memilih modul SDK individual, gunakan AWS SDK for Java bill of materials (BOM) untuk Maven, yang akan memastikan bahwa modul yang Anda tentukan menggunakan versi SDK yang sama dan kompatibel satu sama lain.

Untuk menggunakan BOM, tambahkan `<dependencyManagement>` bagian ke `pom.xml` file aplikasi Anda, tambahkan `aws-java-sdk-bom` sebagai dependensi dan tentukan versi SDK yang ingin Anda gunakan:

```
<dependencyManagement>  
  <dependencies>  
    <dependency>  
      <groupId>com.amazonaws</groupId>  
      <artifactId>aws-java-sdk-bom</artifactId>  
      <version>1.11.1000</version>  
      <type>pom</type>  
      <scope>import</scope>  
    </dependency>  
  </dependencies>  
</dependencyManagement>
```

```
</dependencies>  
</dependencyManagement>
```

Untuk melihat versi terbaru AWS SDK for Java BOM yang tersedia di Maven Central, kunjungi: <https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-bom>. Anda juga dapat menggunakan halaman ini untuk melihat modul (dependensi) mana yang dikelola oleh BOM yang dapat Anda sertakan dalam `<dependencies>` bagian `pom.xml` file proyek Anda.

Sekarang Anda dapat memilih modul individual dari SDK yang Anda gunakan dalam aplikasi Anda. Karena Anda sudah mendeklarasikan versi SDK di BOM, Anda tidak perlu menentukan nomor versi untuk setiap komponen.

```
<dependencies>  
  <dependency>  
    <groupId>com.amazonaws</groupId>  
    <artifactId>aws-java-sdk-s3</artifactId>  
  </dependency>  
  <dependency>  
    <groupId>com.amazonaws</groupId>  
    <artifactId>aws-java-sdk-dynamodb</artifactId>  
  </dependency>  
</dependencies>
```

Anda juga dapat merujuk ke Katalog Kode Sampel AWS untuk mempelajari apa dependensi untuk digunakan untuk diberikan Layanan AWS. Lihat file POM di bawah contoh layanan tertentu. Misalnya, jika Anda tertarik dengan dependensi untuk layanan AWS S3, lihat [contoh lengkapnya](#) GitHub. (Lihatlah pom di bawah `/java/example_code/s3`).

Mengimpor semua modul SDK

Jika Anda ingin menarik seluruh SDK sebagai dependensi, jangan gunakan metode BOM, tetapi cukup mendeklarasikannya `pom.xml` seperti ini:

```
<dependencies>  
  <dependency>  
    <groupId>com.amazonaws</groupId>  
    <artifactId>aws-java-sdk</artifactId>  
    <version>1.11.1000</version>  
  </dependency>  
</dependencies>
```

Bangun proyek Anda

Setelah Anda menyiapkan proyek Anda, Anda dapat membangunnya menggunakan package perintah Maven:

```
mvn package
```

Ini akan membuat `jar` file Anda ditarget direktori.

Membangun SDK dengan Maven

Anda dapat menggunakan Apache Maven untuk membangun SDK dari sumber. Untuk melakukannya, [download kode SDK dari GitHub](#), membongkar secara lokal, dan kemudian jalankan perintah Maven berikut:

```
mvn clean install
```

Menggunakan SDK dengan Gradle

Untuk mengelola dependensi SDK untuk [Gradle](#) proyek, impor Maven BOM untuk AWS SDK for Java ke dalam aplikasi `build.gradle` berkas.

Note

Dalam contoh berikut, ganti `1.12.529` dalam file build dengan versi yang valid dari AWS SDK for Java. Temukan versi terbaru di [Repositori pusat Maven](#).

Penyiapan proyek untuk Gradle 4.6 atau lebih tinggi

[Sejak Gradle 4.6](#), Anda dapat menggunakan fitur dukungan POM Gradle yang ditingkatkan untuk mengimpor file bill of materials (BOM) dengan mendeklarasikan ketergantungan pada BOM.

1. Jika Anda menggunakan Gradle 5.0 atau yang lebih baru, lewati ke langkah 2. Jika tidak, aktifkan `DITINGKATKAN_POM_SUPPORT` fitur di `settings.gradle` berkas.

```
enableFeaturePreview('IMPROVED_POM_SUPPORT')
```

2. Tambahkan BOM ke dependensi bagian dari aplikasi `build.gradle` berkas.

```
...
dependencies {
    implementation platform('com.amazonaws:aws-java-sdk-bom:1.12.529')

    // Declare individual SDK dependencies without version
    ...
}
```

3. Tentukan modul SDK yang akan digunakan di dependensi bagian. Misalnya, berikut ini mencakup ketergantungan untuk Amazon Simple Storage Service (Amazon S3).

```
...
dependencies {
    implementation platform('com.amazonaws:aws-java-sdk-bom:1.12.529')
    implementation 'com.amazonaws:aws-java-sdk-s3'
    ...
}
```

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

Berikut ini adalah contoh lengkap `build.gradle` file yang menyertakan ketergantungan untuk Amazon S3.

```
group 'aws.test'
version '1.0-SNAPSHOT'

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    implementation platform('com.amazonaws:aws-java-sdk-bom:1.12.529')
    implementation 'com.amazonaws:aws-java-sdk-s3'
}
```

Note

Pada contoh sebelumnya, ganti dependensi untuk Amazon S3 dengan dependensi dari AWS Layanan yang akan Anda gunakan dalam proyek Anda. Modul (dependensi) yang dikelola oleh AWS SDK for Java BOM terdaftar di [Repositori pusat Maven](#).

Penyiapan proyek untuk versi Gradle lebih awal dari 4.6

Versi Gradle lebih awal dari 4.6 tidak memiliki dukungan BOM asli. Untuk mengelola AWS SDK for Java dependensi untuk proyek Anda, gunakan Spring [plugin manajemen ketergantungan](#) agar Gradle mengimpor Maven BOM untuk SDK.

1. Tambahkan plugin manajemen ketergantungan ke aplikasi Anda `build.gradle` berkas.

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:1.0.9.RELEASE"
    }
}

apply plugin: "io.spring.dependency-management"
```

2. Tambahkan BOM ke Manajemen Ketergantungan bagian dari file.

```
dependencyManagement {
    imports {
        mavenBom 'com.amazonaws:aws-java-sdk-bom:1.12.529'
    }
}
```

3. Tentukan modul SDK yang akan Anda gunakan di dependensi bagian. Misalnya, berikut ini mencakup ketergantungan untuk Amazon S3.

```
dependencies {
    compile 'com.amazonaws:aws-java-sdk-s3'
}
```

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

Berikut ini adalah contoh lengkap `build.gradle` yang menyertakan ketergantungan untuk Amazon S3.

```
group 'aws.test'
version '1.0'

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:1.0.9.RELEASE"
    }
}

apply plugin: "io.spring.dependency-management"

dependencyManagement {
    imports {
        mavenBom 'com.amazonaws:aws-java-sdk-bom:1.12.529'
    }
}

dependencies {
    compile 'com.amazonaws:aws-java-sdk-s3'
    testCompile group: 'junit', name: 'junit', version: '4.11'
}
```


Note

Pada contoh sebelumnya, ganti dependensi untuk Amazon S3 dengan dependensi dari AWS layanan yang akan Anda gunakan dalam proyek Anda. Modul (dependensi) yang dikelola oleh AWS SDK for Java BOM terdaftar di [Repositori pusat Maven](#).

Untuk informasi selengkapnya tentang menentukan dependensi SDK dengan menggunakan BOM, lihat [Menggunakan SDK dengan Apache Maven](#).

Menyiapkan kredensial AWS sementara dan Wilayah AWS untuk pengembangan

Untuk terhubung ke salah satu layanan yang didukung dengan AWS SDK for Java, Anda harus memberikan kredensial AWS sementara. AWS SDK dan CLI menggunakan rantai penyedia untuk mencari kredensial AWS sementara di sejumlah tempat yang berbeda, termasuk variabel lingkungan sistem/pengguna dan file AWS konfigurasi lokal.

Topik ini memberikan informasi dasar tentang menyiapkan kredensial AWS sementara Anda untuk pengembangan aplikasi lokal menggunakan AWS SDK for Java. Jika Anda perlu menyiapkan kredensial untuk digunakan dalam instans EC2 atau jika Anda menggunakan IDE Eclipse untuk pengembangan, lihat topik berikut sebagai gantinya:

- Saat menggunakan instans EC2, buat peran IAM dan kemudian berikan akses instans EC2 Anda ke peran tersebut seperti yang ditunjukkan dalam [Menggunakan Peran IAM untuk Memberikan Akses ke AWS Sumber Daya aktif Amazon EC2](#).
- Mengatur AWS kredensial dalam Eclipse menggunakan [AWS Toolkit for Eclipse](#). Lihat [Mengatur AWS Kredensial](#) di [Panduan AWS Toolkit for Eclipse Pengguna](#) untuk informasi selengkapnya.

Konfigurasi kredensial sementara

Anda dapat mengkonfigurasi kredensial sementara untuk AWS SDK for Java dalam beberapa cara, tetapi di sini adalah pendekatan yang disarankan:

- Tetapkan kredensial sementara di file profil AWS kredensial di sistem lokal Anda, yang terletak di:
 - `~/.aws/credentials` di Linux macOS, macOS macOS, macOS, macOS, macOS, macOS

- C:\Users\USERNAME\.aws\credentials di Windows

Lihat [the section called “Menyiapkan kredensi sementara untuk SDK”](#) di panduan ini untuk petunjuk tentang cara mendapatkan kredensial sementara Anda.

- Mengatur `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan variabel `AWS_SESSION_TOKEN` lingkungan.

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

```
export AWS_ACCESS_KEY_ID=your_access_key_id
export AWS_SECRET_ACCESS_KEY=your_secret_access_key
export AWS_SESSION_TOKEN=your_session_token
```

Untuk menetapkan variabel ini di Windows, gunakan :

```
set AWS_ACCESS_KEY_ID=your_access_key_id
set AWS_SECRET_ACCESS_KEY=your_secret_access_key
set AWS_SESSION_TOKEN=your_session_token
```

- Untuk instans EC2, tentukan peran IAM dan kemudian berikan akses instans EC2 Anda ke peran itu. Lihat [Peran IAM Amazon EC2](#) di Panduan Amazon EC2 Pengguna untuk Instans Linux untuk diskusi mendetail tentang cara kerjanya.

Setelah Anda menetapkan kredensial AWS sementara Anda menggunakan salah satu metode ini, mereka akan dimuat secara otomatis oleh AWS SDK for Java dengan menggunakan rantai penyedia kredensi default. Untuk informasi lebih lanjut tentang bekerja dengan AWS kredensial dalam aplikasi Java Anda, lihat [Bekerja dengan AWS Kredensial](#).

Kredensial IMDS yang menyegarkan

AWS SDK for Java mendukung keikutsertaan kredensial IMDS yang menyegarkan di latar belakang setiap 1 menit, terlepas dari waktu kedaluwarsa kredensialnya. Ini memungkinkan Anda untuk menyegarkan kredensial lebih sering dan mengurangi kemungkinan bahwa tidak mencapai IMDS memengaruhi AWS ketersediaan yang dirasakan.

```
1. // Refresh credentials using a background thread, automatically every minute. This
   // will log an error if IMDS is down during
2. // a refresh, but your service calls will continue using the cached credentials
   // until the credentials are refreshed
```

```
3. // again one minute later.
4.
5. InstanceProfileCredentialsProvider credentials =
6.     InstanceProfileCredentialsProvider.createAsyncRefreshingProvider(true);
7.
8. AmazonS3Client.builder()
9.     .withCredentials(credentials)
10.    .build();
11.
12. // This is new: When you are done with the credentials provider, you must close it
    to release the background thread.
13. credentials.close();
```

Mengatur Wilayah AWS

Anda harus menetapkan default Wilayah AWS yang akan digunakan untuk mengakses AWS layanan dengan AWS SDK for Java. Untuk kinerja jaringan terbaik, pilih wilayah yang secara geografis dekat dengan Anda (atau pelanggan Anda). Untuk daftar wilayah untuk setiap layanan, lihat [Wilayah dan Titik Akhir](#) di Referensi Amazon Web Services Umum.

Note

Jika Anda tidak memilih wilayah, maka us-east-t-wet-wet-wet-wet-wet-wet-wewet-weweed.

Anda dapat menggunakan teknik serupa untuk menetapkan kredensial untuk mengatur AWS wilayah default Anda:

- Atur file AWS konfigurasi Wilayah AWS di sistem lokal Anda, yang terletak di:
 - `~/.aws/config` di Linux macOS, macOS, macOS, macOS, macOS, macOS, macOS, macOS, Unix
 - `C:\Users\USERNAME\.aws\config` pada Windows

File ini harus berisi baris dalam format berikut:

+

```
[default]
region = your_aws_region
```

+

Ganti yang Anda inginkan Wilayah AWS (misalnya, "us-east-t-wet-1") dengan `your_aws_region`.

- Mengatur variabel `AWS_REGION` lingkungan.

Di Linux, macOS macOS, macOS macOS, macOS, macOS, macOS, macOS, macOS

```
export AWS_REGION=your_aws_region
```

Di Windows, gunakan :

```
set AWS_REGION=your_aws_region
```

Dimana `your_aws_region` adalah Wilayah AWS nama yang diinginkan.

Menggunakan AWS SDK for Java

Bagian ini memberikan informasi umum yang penting tentang pemrograman dengan AWS SDK for Java yang berlaku untuk semua layanan yang mungkin Anda gunakan dengan SDK.

Untuk informasi dan contoh pemrograman khusus layanan (untuk Amazon EC2, Amazon S3, Amazon SWF, dll.), Lihat Contoh [AWS SDK for Java Kode](#).

Topik

- [Praktik Terbaik untuk AWS Pengembangan dengan AWS SDK for Java](#)
- [Membuat Klien Layanan](#)
- [Memberikan kredensi sementara ke AWS SDK for Java](#)
- [Wilayah AWS Seleksi](#)
- [Penanganan Pengecualian](#)
- [Pemrograman Asinkron](#)
- [AWS SDK for Java Panggilan Pencatatan](#)
- [Konfigurasi Klien](#)
- [Kebijakan Kontrol Akses](#)
- [Mengatur JVM TTL untuk pencarian nama DNS](#)
- [Mengaktifkan Metrik untuk AWS SDK for Java](#)

Praktik Terbaik untuk AWS Pengembangan dengan AWS SDK for Java

Praktik terbaik berikut dapat membantu Anda menghindari masalah atau masalah saat Anda mengembangkan AWS aplikasi dengan aplikasi AWS SDK for Java. Kami telah mengatur praktik terbaik berdasarkan layanan.

S3

Hindari ResetExceptions

Saat Anda mengunggah objek Amazon S3 dengan menggunakan aliran (baik melalui AmazonS3 klien atau `TransferManager`), Anda mungkin mengalami masalah konektivitas jaringan atau batas

waktu. Secara default, AWS SDK for Java upaya untuk mencoba kembali transfer yang gagal dengan menandai aliran input sebelum dimulainya transfer dan kemudian mengatur ulang sebelum mencoba lagi.

Jika aliran tidak mendukung tanda dan reset, SDK akan melempar a [ResetException](#) ketika ada kegagalan sementara dan percobaan ulang diaktifkan.

Praktik Terbaik

Kami menyarankan Anda menggunakan aliran yang mendukung menandai dan mengatur ulang operasi.

Cara yang paling dapat diandalkan untuk menghindari a [ResetException](#) adalah dengan menyediakan data dengan menggunakan [File](#) atau [FileInputStream](#), yang AWS SDK for Java dapat ditangani tanpa dibatasi oleh batas tanda dan reset.

Jika aliran bukan [FileInputStream](#) tetapi mendukung tanda dan reset, Anda dapat mengatur batas tanda dengan menggunakan `setReadLimit` metode [RequestClientOptions](#). Nilai defaultnya adalah 128 KB. Menyetel nilai batas baca ke satu byte lebih besar dari ukuran aliran akan secara andal menghindari a [ResetException](#).

Misalnya, jika ukuran maksimum yang diharapkan dari aliran adalah 100.000 byte, atur batas baca menjadi 100,001 (100.000 + 1) byte. Tanda dan reset akan selalu bekerja untuk 100.000 byte atau kurang. Ketahuilah bahwa ini dapat menyebabkan beberapa aliran menyangga jumlah byte itu ke dalam memori.

Membuat Klien Layanan

Untuk membuat permintaan Amazon Web Services, pertama-tama Anda membuat objek klien layanan. Cara yang disarankan adalah dengan menggunakan pembuat klien layanan.

Masing-masing Layanan AWS memiliki antarmuka layanan dengan metode untuk setiap tindakan di API layanan. [Misalnya, antarmuka layanan untuk DynamoDB bernama `DbClient`. `AmazonDynamo`](#) Setiap antarmuka layanan memiliki pembangun klien yang sesuai yang dapat Anda gunakan untuk membangun implementasi antarmuka layanan. Kelas pembangun klien bernama [`AmazonDynamoDBClientBuilder`](#). DynamoDB

Memperoleh Client Builder

Untuk mendapatkan instance dari pembuat klien, gunakan metode pabrik statis `standard`, seperti yang ditunjukkan pada contoh berikut.

```
AmazonDynamoDBClientBuilder builder = AmazonDynamoDBClientBuilder.standard();
```

Setelah Anda memiliki pembangun, Anda dapat menyesuaikan properti klien dengan menggunakan banyak penyetel fasih di API pembangun. Misalnya, Anda dapat mengatur wilayah kustom dan penyedia kredensial kustom, sebagai berikut.

```
AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCredentials(new ProfileCredentialsProvider("myProfile"))
    .build();
```

Note

`withXXX` metode fasih mengembalikan `builder` objek sehingga Anda dapat menghubungkan panggilan metode untuk kenyamanan dan untuk kode yang lebih mudah dibaca. Setelah Anda mengkonfigurasi properti yang Anda inginkan, Anda dapat memanggil `build` metode untuk membuat klien. Setelah klien dibuat, itu tidak dapat diubah dan panggilan apa pun ke `setRegion` atau `setEndpoint` akan gagal.

Pembangun dapat membuat beberapa klien dengan konfigurasi yang sama. Saat Anda menulis aplikasi, ketahuilah bahwa pembuatnya bisa berubah dan tidak aman untuk utas.

Kode berikut menggunakan builder sebagai pabrik untuk instance klien.

```
public class DynamoDBClientFactory {
    private final AmazonDynamoDBClientBuilder builder =
        AmazonDynamoDBClientBuilder.standard()
            .withRegion(Regions.US_WEST_2)
            .withCredentials(new ProfileCredentialsProvider("myProfile"));

    public AmazonDynamoDB createClient() {
        return builder.build();
    }
}
```

[Pembangun juga mengekspos setter fasih untuk ClientConfiguration dan RequestMetricCollector, dan daftar kustom 2. RequestHandler](#)

Berikut ini adalah contoh lengkap yang mengesampingkan semua properti yang dapat dikonfigurasi.

```
AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCredentials(new ProfileCredentialsProvider("myProfile"))
    .withClientConfiguration(new ClientConfiguration().withRequestTimeout(5000))
    .withMetricsCollector(new MyCustomMetricsCollector())
    .withRequestHandlers(new MyCustomRequestHandler(), new
MyOtherCustomRequestHandler)
    .build();
```

Membuat Klien Async

Ini AWS SDK for Java memiliki klien asinkron (atau asinkron) untuk setiap layanan (kecuali untuk Amazon S3), dan pembuat klien asinkron yang sesuai untuk setiap layanan.

Untuk membuat klien DynamoDB async dengan default ExecutorService

```
AmazonDynamoDBAsync ddbAsync = AmazonDynamoDBAsyncClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCredentials(new ProfileCredentialsProvider("myProfile"))
    .build();
```

Selain opsi konfigurasi yang didukung oleh pembuat klien sinkron (atau sinkronisasi), klien asinkron memungkinkan Anda menyetel kustom [ExecutorFactory](#) untuk mengubah yang digunakan klien `ExecutorService` asinkron. `ExecutorFactory` adalah antarmuka fungsional, sehingga berinteraksi dengan ekspresi lambda Java 8 dan referensi metode.

Untuk membuat klien async dengan eksekutor kustom

```
AmazonDynamoDBAsync ddbAsync = AmazonDynamoDBAsyncClientBuilder.standard()
    .withExecutorFactory(() -> Executors.newFixedThreadPool(10))
    .build();
```

Menggunakan DefaultClient

Baik pembuat klien sinkronisasi dan asinkron memiliki metode pabrik lain bernama. `defaultClient` Metode ini membuat klien layanan dengan konfigurasi default, menggunakan rantai penyedia default

untuk memuat kredensyal dan file. Wilayah AWS Jika kredensyal atau wilayah tidak dapat ditentukan dari lingkungan tempat aplikasi berjalan, panggilan ke `defaultClient` gagal. Lihat [Bekerja dengan AWS Kredensyal](#) dan [Wilayah AWS Seleksi](#) untuk informasi selengkapnya tentang bagaimana kredensyal dan wilayah ditentukan.

Untuk membuat klien layanan default

```
AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();
```

Siklus Hidup Klien

Klien layanan di SDK aman untuk benang dan, untuk kinerja terbaik, Anda harus memperlakukannya sebagai objek yang berumur panjang. Setiap klien memiliki sumber daya kolam koneksi sendiri. Secara eksplisit menutup klien ketika mereka tidak lagi diperlukan untuk menghindari kebocoran sumber daya.

Untuk secara eksplisit mematikan klien, panggil metode `shutdown`. Setelah menelepon `shutdown`, semua sumber daya klien dilepaskan dan klien tidak dapat digunakan.

Untuk mematikan klien

```
AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();  
ddb.shutdown();  
// Client is now unusable
```

Memberikan kredensi sementara ke AWS SDK for Java

Untuk membuat permintaan Amazon Web Services, Anda harus memberikan kredensi AWS sementara AWS SDK for Java untuk digunakan saat memanggil layanan. Anda dapat melakukan ini dengan cara berikut:

- Gunakan rantai penyedia kredensyal default (disarankan).
- Gunakan penyedia kredensyal atau rantai penyedia tertentu (atau buat sendiri).
- Berikan sendiri kredensi sementara dalam kode.

Menggunakan Rantai Penyedia Kredensyal Default

[Saat Anda menginisialisasi klien layanan baru tanpa memberikan argumen apa pun, AWS SDK for Java upaya untuk menemukan kredensyal sementara dengan menggunakan rantai penyedia kredensyal default yang diimplementasikan oleh kelas `Default.AWSCredentialsProviderChain`](#) Rantai penyedia kredensyal default mencari kredensyal dalam urutan ini:

1. Variabel lingkungan -`AWS_ACCESS_KEY_ID`,`AWS_SECRET_ACCESS_KEY`, dan`AWS_SESSION_TOKEN`. AWS SDK for Java Menggunakan [`EnvironmentVariableCredentialsProvider`](#) kelas untuk memuat kredensyal ini.
2. Properti sistem Java -`aws.accessKeyId`,`aws.secretKey`, dan`aws.sessionToken`. AWS SDK for Java Kegunaan [`SystemPropertiesCredentialsProvider`](#) untuk memuat kredensyal ini.
3. Kredensyal Token Identitas Web dari lingkungan atau wadah.
4. File profil kredensyal default - biasanya terletak di `~/.aws/credentials` (lokasi dapat bervariasi per platform), dan dibagikan oleh banyak AWS SDK dan oleh. AWS CLI AWS SDK for Java Kegunaan [`ProfileCredentialsProvider`](#) untuk memuat kredensyal ini.

Anda dapat membuat file kredensyal dengan menggunakan `aws configure` perintah yang disediakan oleh AWS CLI, atau Anda dapat membuatnya dengan mengedit file dengan editor teks. Untuk informasi tentang format file kredensyal, lihat Format File [AWS Kredensial](#).

5. Kredensyal kontainer Amazon ECS - dimuat dari Amazon ECS jika variabel lingkungan disetel. `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` AWS SDK for Java Kegunaan [`ContainerCredentialsProvider`](#) untuk memuat kredensyal ini. Anda dapat menentukan alamat IP untuk nilai ini.
6. Instance profile credentials - digunakan pada instans EC2, dan dikirimkan melalui layanan metadata. Amazon EC2 AWS SDK for Java Kegunaan [`InstanceProfileCredentialsProvider`](#) untuk memuat kredensyal ini. Anda dapat menentukan alamat IP untuk nilai ini.

Note

Kredensyal profil instance hanya digunakan jika tidak `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` disetel. Lihat [EC2 `ContainerCredentialsProviderWrapper`](#) untuk informasi lebih lanjut.

Tetapkan kredensial sementara

Untuk dapat menggunakan kredensial AWS sementara, mereka harus ditetapkan setidaknya di salah satu lokasi sebelumnya. Untuk informasi tentang menyetel kredensial, lihat topik berikut:

- Untuk menentukan kredensial di lingkungan atau dalam file profil kredensial default, lihat [the section called “Konfigurasi kredensial sementara”](#)
- Untuk mengatur properti sistem Java, lihat tutorial [System Properties](#) di situs web resmi Java Tutorial.
- Untuk menyiapkan dan menggunakan kredensial profil instans dengan instans EC2 Anda, lihat [Menggunakan Peran IAM untuk Memberikan Akses ke Sumber Daya pada](#). AWS Amazon EC2

Menetapkan profil kredensial alternatif

AWS SDK for Java Menggunakan profil default secara default, tetapi ada cara untuk menyesuaikan profil mana yang bersumber dari file kredensial.

Anda dapat menggunakan variabel lingkungan AWS Profil untuk mengubah profil yang dimuat oleh SDK.

Misalnya, di Linux, macOS, atau Unix Anda akan menjalankan perintah berikut untuk mengubah profil ke MyProfile.

```
export AWS_PROFILE="myProfile"
```

Di Windows Anda akan menggunakan yang berikut ini.

```
set AWS_PROFILE="myProfile"
```

Menyetel variabel `AWS_PROFILE` lingkungan memengaruhi pemuatan kredensial untuk semua AWS SDK dan Alat yang didukung secara resmi (termasuk AWS CLI dan). AWS Tools for Windows PowerShell Untuk mengubah hanya profil untuk aplikasi Java, Anda dapat menggunakan properti sistem `aws.profile` sebagai gantinya.

Note

Variabel lingkungan lebih diutamakan daripada properti sistem.

Menetapkan lokasi file kredensial alternatif

AWS SDK for Java Memuat kredensial AWS sementara secara otomatis dari lokasi file kredensial default. Namun, Anda juga dapat menentukan lokasi dengan menyetel variabel `AWS_CREDENTIAL_PROFILES_FILE` lingkungan dengan jalur lengkap ke file kredensial.

Anda dapat menggunakan fitur ini untuk sementara mengubah lokasi di mana AWS SDK for Java mencari file kredensial Anda (misalnya, dengan mengatur variabel ini dengan baris perintah). Atau Anda dapat mengatur variabel lingkungan di lingkungan pengguna atau sistem Anda untuk mengubahnya untuk pengguna atau seluruh sistem.

Untuk mengganti lokasi file kredensial default

- Atur variabel `AWS_CREDENTIAL_PROFILES_FILE` lingkungan ke lokasi file AWS kredensial Anda.
 - Di Linux, macOS, atau Unix, gunakan:

```
export AWS_CREDENTIAL_PROFILES_FILE=path/to/credentials_file
```

- Di Windows, gunakan:

```
set AWS_CREDENTIAL_PROFILES_FILE=path/to/credentials_file
```

Credentials format berkas

Dengan mengikuti [petunjuk dalam pengaturan Dasar](#) panduan ini, file kredensial Anda harus memiliki format dasar berikut.

```
[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>

[profile2]
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>
```

Nama profil ditentukan dalam tanda kurung siku (misalnya, `[default]`), diikuti oleh bidang yang dapat dikonfigurasi di profil itu sebagai pasangan nilai kunci. Anda dapat memiliki beberapa profil di

credentials file Anda, yang dapat ditambahkan atau diedit menggunakan `aws configure --profile PROFILE_NAME` untuk memilih profil yang akan dikonfigurasi.

Anda dapat menentukan bidang tambahan, seperti `metadata_service_timeout`, dan `metadata_service_num_attempts`. Ini tidak dapat dikonfigurasi dengan CLI—Anda harus mengedit file dengan tangan jika Anda ingin menggunakannya. Untuk informasi selengkapnya tentang file konfigurasi dan bidangnya yang tersedia, lihat [Mengonfigurasi AWS Command Line Interface](#) dalam Panduan AWS Command Line Interface Pengguna.

Memuat kredensial

Setelah Anda menyetel kredensial sementara, SDK akan memuatnya dengan menggunakan rantai penyedia kredensial default.

Untuk melakukan ini, Anda membuat instance Layanan AWS klien tanpa secara eksplisit memberikan kredensial kepada pembangun, sebagai berikut.

```
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();
```

Tentukan penyedia kredensial atau rantai penyedia

Anda dapat menentukan penyedia kredensial yang berbeda dari rantai penyedia kredensial default dengan menggunakan pembuat klien.

Anda memberikan instance penyedia kredensial atau rantai penyedia ke pembuat klien yang mengambil [AWSCredentialsProvider](#) antarmuka sebagai input. Contoh berikut menunjukkan cara menggunakan kredensial lingkungan secara khusus.

```
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withCredentials(new EnvironmentVariableCredentialsProvider())
    .build();
```

Untuk daftar lengkap penyedia kredensial AWS SDK for Java yang disediakan dan rantai penyedia, lihat Semua Kelas Penerapan yang Dikenal di [AWSCredentialsProvider](#)

Note

Anda dapat menggunakan teknik ini untuk menyediakan penyedia kredensi atau rantai penyedia yang Anda buat dengan menggunakan penyedia kredensial Anda sendiri yang mengimplementasikan `AWSCredentialsProvider` antarmuka, atau dengan mensubklasifikasikan kelas. [AWSCredentialsProviderChain](#)

Secara eksplisit menentukan kredensial sementara

Jika rantai kredensial default atau penyedia atau rantai penyedia khusus atau khusus tidak berfungsi untuk kode Anda, Anda dapat menyetel kredensial yang Anda berikan secara eksplisit. Jika Anda telah mengambil kredensial sementara menggunakan AWS STS, gunakan metode ini untuk menentukan kredensial untuk akses. AWS

1. Buat instance [BasicSessionCredentials](#) kelas, dan berikan kunci AWS akses, kunci AWS rahasia, dan token AWS sesi yang akan digunakan SDK untuk koneksi.
2. Buat [AWSStaticCredentialsProvider](#) dengan `AWSCredentials` objek.
3. Konfigurasi pembuat klien dengan `AWSStaticCredentialsProvider` dan bangun klien.

Berikut adalah contohnya.

```
BasicSessionCredentials awsCreds = new BasicSessionCredentials("access_key_id",
    "secret_key_id", "session_token");
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
    .build();
```

Info Selengkapnya

- [Mendaftar AWS dan Membuat Pengguna IAM](#)
- [Menyiapkan AWS Kredensial dan Wilayah untuk Pembangunan](#)
- [Menggunakan Peran IAM untuk Memberikan Akses ke AWS Sumber Daya di Amazon EC2](#)

Wilayah AWS Seleksi

Wilayah memungkinkan Anda mengakses AWS layanan yang secara fisik berada di wilayah geografis tertentu. Ini dapat berguna baik untuk redundansi dan untuk menjaga data dan aplikasi Anda berjalan dekat dengan tempat Anda dan pengguna Anda akan mengaksesnya.

Memeriksa Ketersediaan Layanan di Wilayah

Untuk melihat apakah tertentu Layanan AWS tersedia di suatu wilayah, gunakan `isServiceSupported` metode di wilayah yang ingin Anda gunakan.

```
Region.getRegion(Regions.US_WEST_2)
    .isServiceSupported(AmazonDynamoDB.ENDPOINT_PREFIX);
```

Lihat dokumentasi kelas [Regions](#) untuk wilayah yang dapat Anda tentukan, dan gunakan awalan titik akhir layanan untuk melakukan kueri. Awalan endpoint setiap layanan didefinisikan dalam antarmuka layanan. [Misalnya, awalan DynamoDB endpoint didefinisikan dalam AmazonDynamoDB.](#)

Memilih Wilayah

Dimulai dengan versi 1.4 AWS SDK for Java, Anda dapat menentukan nama wilayah dan SDK akan secara otomatis memilih titik akhir yang sesuai untuk Anda. Untuk memilih endpoint sendiri, lihat [Memilih Endpoint Spesifik](#).

Untuk menetapkan wilayah secara eksplisit, sebaiknya gunakan enum [Regions](#). Ini adalah enumerasi dari semua wilayah yang tersedia untuk umum. Untuk membuat klien dengan wilayah dari enum, gunakan kode berikut.

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();
```

Jika wilayah yang Anda coba gunakan tidak ada di `Regions` enum, Anda dapat mengatur wilayah menggunakan string yang mewakili nama wilayah.

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()
    .withRegion("{region_api_default}")
    .build();
```

Note

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

Memilih Endpoint Tertentu

Setiap AWS klien dapat dikonfigurasi untuk menggunakan titik akhir tertentu dalam suatu wilayah dengan memanggil `withEndpointConfiguration` metode saat membuat klien.

Misalnya, untuk mengkonfigurasi Amazon S3 klien untuk menggunakan Wilayah Eropa (Irlandia), gunakan kode berikut.

```
AmazonS3 s3 = AmazonS3ClientBuilder.standard()
    .withEndpointConfiguration(new EndpointConfiguration(
        "https://s3.eu-west-1.amazonaws.com",
        "eu-west-1"))
    .withCredentials(CREDENTIALS_PROVIDER)
    .build();
```

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

Secara Otomatis Menentukan Wilayah dari Lingkungan

Important

Bagian ini hanya berlaku ketika menggunakan [pembuat klien](#) untuk mengakses AWS layanan. AWS klien yang dibuat dengan menggunakan konstruktor klien tidak akan secara otomatis menentukan wilayah dari lingkungan dan akan, sebaliknya, menggunakan wilayah SDK default (`useAST1`).

Saat menjalankan on Amazon EC2 atau Lambda, Anda mungkin ingin mengonfigurasi klien untuk menggunakan wilayah yang sama dengan tempat kode Anda berjalan. Ini memisahkan kode Anda dari lingkungan tempat ia berjalan dan membuatnya lebih mudah untuk menerapkan aplikasi Anda ke beberapa wilayah untuk latensi atau redundansi yang lebih rendah.

Anda harus menggunakan pembuat klien agar SDK secara otomatis mendeteksi wilayah tempat kode Anda berjalan.

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

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();
```

Ini sama dengan menggunakan `standard` diikuti oleh `build`.

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()  
    .build();
```

Jika Anda tidak secara eksplisit menyetel wilayah menggunakan `withRegion` metode, SDK akan berkonsultasi dengan rantai penyedia wilayah default untuk mencoba dan menentukan wilayah yang akan digunakan.

Rantai Penyedia Wilayah Default

Berikut ini adalah proses pencarian wilayah:

1. Wilayah eksplisit apa pun yang disetel dengan menggunakan `withRegion` atau `setRegion` pada pembangun itu sendiri lebih diutamakan daripada yang lain.
2. Variabel `AWS_REGION` lingkungan diperiksa. Jika disetel, wilayah itu digunakan untuk mengkonfigurasi klien.

Note

Variabel lingkungan ini diatur oleh Lambda wadah.

3. SDK memeriksa file konfigurasi AWS bersama (biasanya terletak di `~/ .aws/config`). Jika properti `region` ada, SDK menggunakannya.
 - 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 menyesuaikan profil yang dimuat oleh SDK.
4. SDK mencoba menggunakan layanan metadata Amazon EC2 instance untuk menentukan wilayah instance yang sedang berjalan. Amazon EC2

5. Jika SDK masih belum menemukan wilayah pada saat ini, pembuatan klien gagal dengan pengecualian.

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

Penanganan Pengecualian

Memahami bagaimana dan kapan pengecualian AWS SDK for Java 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 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.

Untuk informasi selengkapnya tentang penggunaan pengecualian yang dicentang dan tidak dicentang, lihat:

- [Pengecualian yang Tidak Dicentang — Kontroversi](#)
- [Masalah dengan Pengecualian yang Diperiksa](#)
- [Pengecualian Java yang diperiksa adalah kesalahan \(dan inilah yang ingin saya lakukan tentang hal itu\)](#)

AmazonServiceException (dan Subclass)

[AmazonServiceException](#) adalah pengecualian paling umum yang akan Anda alami saat menggunakan AWS SDK for Java. Pengecualian ini merupakan respons kesalahan dari file Layanan AWS. Misalnya, jika Anda mencoba menghentikan Amazon EC2 instance yang tidak ada, EC2 akan mengembalikan respons kesalahan dan semua detail respons kesalahan itu akan disertakan dalam `AmazonServiceException` yang dilemparkan. Untuk beberapa kasus, subclass dilemparkan untuk memungkinkan pengembang mengontrol secara halus atas penanganan kasus kesalahan melalui blok catch. `AmazonServiceException`

Ketika Anda menemukan `AmazonServiceException`, 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.

`AmazonServiceException` memberi Anda informasi seperti:

- Kode status HTTP yang dikembalikan
- Kode AWS kesalahan yang dikembalikan
- Pesan kesalahan terperinci dari layanan
- AWS ID permintaan untuk permintaan yang gagal

`AmazonServiceException` juga mencakup informasi tentang apakah permintaan yang gagal adalah kesalahan pemanggil (permintaan dengan nilai ilegal) atau kesalahan (kesalahan layanan internal). Layanan AWS

AmazonClientException

[AmazonClientException](#) menunjukkan bahwa masalah terjadi di dalam kode klien Java, baik saat mencoba mengirim permintaan ke AWS atau saat mencoba mengurai respons dari AWS. `AmazonClientException` umumnya lebih parah daripada `AmazonServiceException`, dan menunjukkan masalah besar yang mencegah klien melakukan panggilan layanan ke AWS layanan. Misalnya, AWS SDK for Java melempar `AmazonClientException` jika tidak ada koneksi jaringan yang tersedia ketika Anda mencoba memanggil operasi pada salah satu klien.

Pemrograman Asinkron

Anda dapat menggunakan metode sinkron atau asinkron untuk memanggil operasi pada layanan. AWS 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. AWS SDK for Java Ini menyediakan dua cara: objek masa depan dan metode callback.

Java Berjangka

Metode asinkron dalam AWS SDK for Java mengembalikan objek [Future](#) yang berisi hasil operasi asinkron di masa depan.

Panggil `Future isDone()` metode untuk melihat apakah layanan telah menyediakan objek respons. Ketika respon sudah siap, Anda bisa mendapatkan objek respon dengan memanggil `Future get()` metode. Anda dapat menggunakan mekanisme ini untuk melakukan polling secara berkala untuk hasil operasi asinkron sementara aplikasi Anda terus bekerja pada hal-hal lain.

Berikut adalah contoh operasi asinkron yang memanggil Lambda fungsi, menerima `Future` yang dapat menampung objek. [InvokeResult](#) `InvokeResult`Objek diambil hanya setelah `isDone()` itu `true`.

```
import com.amazonaws.services.lambda.AWSLambdaAsyncClient;
import com.amazonaws.services.lambda.model.InvokeRequest;
import com.amazonaws.services.lambda.model.InvokeResult;
import java.nio.ByteBuffer;
import java.util.concurrent.Future;
import java.util.concurrent.ExecutionException;

public class InvokeLambdaFunctionAsync
{
    public static void main(String[] args)
    {
        String function_name = "HelloFunction";
        String function_input = "{\"who\":\"SDK for Java\"}";

        AWSLambdaAsync lambda = AWSLambdaAsyncClientBuilder.defaultClient();
        InvokeRequest req = new InvokeRequest()
            .withFunctionName(function_name)
            .withPayload(ByteBuffer.wrap(function_input.getBytes()));

        Future<InvokeResult> future_res = lambda.invokeAsync(req);
    }
}
```

```
System.out.print("Waiting for future");
while (future_res.isDone() == false) {
    System.out.print(".");
    try {
        Thread.sleep(1000);
    }
    catch (InterruptedException e) {
        System.err.println("\nThread.sleep() was interrupted!");
        System.exit(1);
    }
}

try {
    InvokeResult res = future_res.get();
    if (res.getStatusCode() == 200) {
        System.out.println("\nLambda function returned:");
        ByteBuffer response_payload = res.getPayload();
        System.out.println(new String(response_payload.array()));
    }
    else {
        System.out.format("Received a non-OK response from {AWS}: %d\n",
            res.getStatusCode());
    }
}
catch (InterruptedException | ExecutionException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

System.exit(0);
}
```

Callback Asinkron

Selain menggunakan Future objek Java untuk memantau status permintaan asinkron, SDK juga memungkinkan Anda untuk mengimplementasikan kelas yang menggunakan antarmuka [AsyncHandler](#). AsyncHandler menyediakan dua metode yang dipanggil tergantung pada bagaimana permintaan selesai: onSuccess dan onError.

Keuntungan utama dari pendekatan antarmuka callback adalah membebaskan Anda dari keharusan melakukan polling Future objek untuk mengetahui kapan permintaan telah selesai. Sebagai

gantinya, kode Anda dapat segera memulai aktivitas berikutnya, dan mengandalkan SDK untuk memanggil handler Anda pada waktu yang tepat.

```
import com.amazonaws.services.lambda.AWSLambdaAsync;
import com.amazonaws.services.lambda.AWSLambdaAsyncClientBuilder;
import com.amazonaws.services.lambda.model.InvokeRequest;
import com.amazonaws.services.lambda.model.InvokeResult;
import com.amazonaws.handlers.AsyncHandler;
import java.nio.ByteBuffer;
import java.util.concurrent.Future;

public class InvokeLambdaFunctionCallback
{
    private class AsyncLambdaHandler implements AsyncHandler<InvokeRequest,
    InvokeResult>
    {
        public void onSuccess(InvokeRequest req, InvokeResult res) {
            System.out.println("\nLambda function returned:");
            ByteBuffer response_payload = res.getPayload();
            System.out.println(new String(response_payload.array()));
            System.exit(0);
        }

        public void onError(Exception e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void main(String[] args)
    {
        String function_name = "HelloFunction";
        String function_input = "{\\"who\\":\\"SDK for Java\\"}";

        AWSLambdaAsync lambda = AWSLambdaAsyncClientBuilder.defaultClient();
        InvokeRequest req = new InvokeRequest()
            .withFunctionName(function_name)
            .withPayload(ByteBuffer.wrap(function_input.getBytes()));

        Future<InvokeResult> future_res = lambda.invokeAsync(req, new
        AsyncLambdaHandler());

        System.out.print("Waiting for async callback");
    }
}
```

```
while (!future_res.isDone() && !future_res.isCancelled()) {
    // perform some other tasks...
    try {
        Thread.sleep(1000);
    }
    catch (InterruptedException e) {
        System.err.println("Thread.sleep() was interrupted!");
        System.exit(0);
    }
    System.out.print(".");
}
}
```

Praktik Terbaik

Eksekusi Callback

Implementasi Anda `AsyncHandler` dijalankan di dalam kumpulan utas yang dimiliki oleh klien asinkron. Kode pendek dan cepat dieksekusi paling tepat di dalam `AsyncHandler` implementasi Anda. Kode yang berjalan lama atau memblokir di dalam metode handler Anda dapat menyebabkan pertentangan untuk kumpulan utas yang digunakan oleh klien asinkron, dan dapat mencegah klien mengeksekusi permintaan. Jika Anda memiliki tugas yang berjalan lama yang perlu dimulai dari callback, minta callback menjalankan tugasnya di thread baru atau di kumpulan utas yang dikelola oleh aplikasi Anda.

Konfigurasi Kolam Benang

Klien asinkron di AWS SDK for Java menyediakan kumpulan utas default yang seharusnya berfungsi untuk sebagian besar aplikasi. Anda dapat menerapkan kustom [ExecutorService](#) dan meneruskannya ke klien AWS SDK for Java asinkron untuk kontrol lebih besar atas bagaimana kumpulan utas dikelola.

Misalnya, Anda dapat memberikan `ExecutorService` implementasi yang menggunakan kustom [ThreadFactory](#) untuk mengontrol cara nama thread di pool, atau untuk mencatat informasi tambahan tentang penggunaan thread.

Akses Asinkron

[TransferManager](#) Kelas di SDK menawarkan dukungan asinkron untuk bekerja dengan Amazon S3 `TransferManager` mengelola unggahan dan unduhan asinkron, menyediakan pelaporan kemajuan mendetail tentang transfer, dan mendukung panggilan balik ke berbagai peristiwa.

AWS SDK for Java Panggilan Pencatatan

AWS SDK for Java Ini diinstrumentasi dengan [Apache Commons Logging](#), 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, antara lain. Topik ini menunjukkan cara menggunakan Log4j. Anda dapat menggunakan fungsionalitas logging SDK tanpa membuat perubahan apa pun pada kode aplikasi Anda.

Untuk mempelajari lebih lanjut tentang [Log4j](#), lihat situs web [Apache](#).

Note

Topik ini berfokus pada Log4j 1.x. Log4j2 tidak secara langsung mendukung Apache Commons Logging, tetapi menyediakan adaptor yang mengarahkan panggilan logging secara otomatis ke Log4j2 menggunakan antarmuka Apache Commons Logging. Untuk informasi selengkapnya, lihat [Commons Logging Bridge di dokumentasi Log4j2](#).

Unduh Log4J JAR

Untuk menggunakan Log4j dengan SDK, Anda perlu mengunduh Log4j JAR dari situs web Apache. SDK tidak termasuk JAR. Salin file JAR ke lokasi yang ada di classpath Anda.

Log4j menggunakan file konfigurasi, `log4j.properties`. Contoh file konfigurasi ditunjukkan di bawah ini. Salin file konfigurasi ini ke direktori di classpath Anda. Log4j JAR dan file `log4j.properties` tidak harus berada di direktori yang sama.

[File konfigurasi `log4j.properties` menentukan properti seperti tingkat logging, di mana output logging dikirim \(misalnya, ke file atau ke konsol\), dan format output.](#) Level logging adalah granularitas output yang dihasilkan logger. Log4j mendukung konsep beberapa hierarki logging. Level logging diatur secara independen untuk setiap hierarki. Dua hierarki logging berikut tersedia di AWS SDK for Java:

- log4j.logger.com.amazonaws
- log4j.logger.org.apache.http.wire

Mengatur Classpath

Baik Log4j JAR dan file log4j.properties harus terletak di classpath Anda. Jika Anda menggunakan [Apache Ant](#), atur classpath di path elemen dalam file Ant Anda. Contoh berikut menunjukkan elemen path dari file Ant untuk Amazon S3 [contoh](#) yang disertakan dengan SDK.

```
<path id="aws.java.sdk.classpath">
  <fileset dir="../../third-party" includes="**/*.jar"/>
  <fileset dir="../../lib" includes="**/*.jar"/>
  <pathelement location="."/>
</path>
```

Jika Anda menggunakan Eclipse IDE, Anda dapat mengatur classpath dengan membuka menu dan menavigasi ke Project | Properties | Java Build Path.

Kesalahan dan Peringatan Khusus Layanan

Sebaiknya Anda selalu membiarkan hierarki logger “com.amazonaws” disetel ke “WARN” untuk menangkap pesan penting apa pun dari pustaka klien. Misalnya, jika Amazon S3 klien 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.

File log4j.properties berikut menyetel `rootLogger` ke `WARN`, yang menyebabkan pesan peringatan dan kesalahan dari semua logger dalam hierarki “com.amazonaws” disertakan. Atau, Anda dapat secara eksplisit mengatur logger `com.amazonaws` ke `WARN`.

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
# Or you can explicitly enable WARN and ERROR messages for the {AWS} Java clients
log4j.logger.com.amazonaws=WARN
```

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. AWS ID permintaan dapat diakses secara terprogram melalui objek Exception di SDK untuk panggilan layanan yang gagal, dan juga dapat dilaporkan melalui tingkat log DEBUG di logger “com.amazonaws.request”.

File log4j.properties berikut memungkinkan ringkasan permintaan dan tanggapan, termasuk ID permintaan. AWS

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
# Turn on DEBUG logging in com.amazonaws.request to log
# a summary of requests/responses with {AWS} request IDs
log4j.logger.com.amazonaws.request=DEBUG
```

Berikut adalah contoh output log.

```
2009-12-17 09:53:04,269 [main] DEBUG com.amazonaws.request - Sending
Request: POST https://rds.amazonaws.com / Parameters: (MaxRecords: 20,
Action: DescribeEngineDefaultParameters, SignatureMethod: HmacSHA256,
AWSAccessKeyId: ACCESSKEYID, Version: 2009-10-16, SignatureVersion: 2,
Engine: mysql5.1, Timestamp: 2009-12-17T17:53:04.267Z, Signature:
q963XH63Lcov15Rr71AP1zlye99rmWwT9DfuQaNznkD, ) 2009-12-17 09:53:04,464
[main] DEBUG com.amazonaws.request - Received successful response: 200, {AWS}
Request ID: 694d1242-cee0-c85e-f31f-5dab1ea18bc6 2009-12-17 09:53:04,469
[main] DEBUG com.amazonaws.request - Sending Request: POST
https://rds.amazonaws.com / Parameters: (ResetAllParameters: true, Action:
ResetDBParameterGroup, SignatureMethod: HmacSHA256, DBParameterGroupName:
java-integ-test-param-group-00000000000000, AWSAccessKeyId: ACCESSKEYID,
Version: 2009-10-16, SignatureVersion: 2, Timestamp:
2009-12-17T17:53:04.467Z, Signature:
9WcgfPwTobvLVcpyhbrdN7P713uH0oviYQ4yZ+TQjsQ=, )

2009-12-17 09:53:04,646 [main] DEBUG com.amazonaws.request - Received
successful response: 200, {AWS} Request ID:
694d1242-cee0-c85e-f31f-5dab1ea18bc6
```

Penebangan Kawat Verbose

Dalam beberapa kasus, akan berguna untuk melihat permintaan dan tanggapan yang tepat yang AWS SDK for Java dikirim dan diterima. Anda tidak boleh mengaktifkan pencatatan ini di sistem produksi karena menulis permintaan besar (misalnya, file yang diunggah Amazon S3) atau tanggapan dapat memperlambat aplikasi secara signifikan. Jika Anda benar-benar membutuhkan akses ke informasi ini, Anda dapat mengaktifkannya sementara melalui logger Apache HttpClient 4. Mengaktifkan tingkat DEBUG pada `org.apache.http.wire` logger memungkinkan pencatatan untuk semua data permintaan dan respons.

File `log4j.properties` berikut mengaktifkan full wire logging di Apache HttpClient 4 dan seharusnya hanya dihidupkan sementara karena dapat memiliki dampak kinerja yang signifikan pada aplikasi Anda.

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
# Log all HTTP content (headers, parameters, content, etc) for
# all requests and responses. Use caution with this since it can
# be very expensive to log such verbose data!
log4j.logger.org.apache.http.wire=DEBUG
```

Pencatatan Metrik Latensi

Jika Anda memecahkan masalah dan ingin melihat metrik seperti proses mana yang paling memakan waktu atau apakah sisi server atau klien memiliki latensi yang lebih besar, logger latensi dapat membantu. Setel `com.amazonaws.latency` logger ke DEBUG untuk mengaktifkan logger ini.

Note

Logger ini hanya tersedia jika metrik SDK diaktifkan. Untuk mempelajari lebih lanjut tentang paket metrik SDK, lihat [Mengaktifkan Metrik](#) untuk paket. AWS SDK for Java

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
```

```
log4j.logger.com.amazonaws.latency=DEBUG
```

Berikut adalah contoh output log.

```
com.amazonaws.latency - ServiceName=[{S3}], StatusCode=[200],
ServiceEndpoint=[https://list-objects-integ-test-test.s3.amazonaws.com],
RequestType=[ListObjectsV2Request], AWSRequestID=[REQUESTID],
HttpClientPoolPendingCount=0,
RetryCapacityConsumed=0, HttpClientPoolAvailableCount=0, RequestCount=1,
HttpClientPoolLeasedCount=0, ResponseProcessingTime=[52.154],
ClientExecuteTime=[487.041],
HttpClientSendRequestTime=[192.931], HttpRequestTime=[431.652],
RequestSigningTime=[0.357],
CredentialsRequestTime=[0.011, 0.001], HttpClientReceiveResponseTime=[146.272]
```

Konfigurasi Klien

AWS SDK for Java Ini memungkinkan Anda untuk mengubah konfigurasi klien default, yang sangat membantu ketika Anda ingin:

- Connect ke Internet melalui proxy
- Ubah pengaturan transport HTTP, seperti batas waktu koneksi dan permintaan percobaan ulang
- Tentukan petunjuk ukuran buffer soket TCP

Konfigurasi Proxy

Saat membuat objek klien, Anda dapat meneruskan [ClientConfiguration](#) objek opsional untuk menyesuaikan konfigurasi klien.

Jika Anda terhubung ke Internet melalui server proxy, Anda harus mengonfigurasi pengaturan server proxy Anda (host proxy, port, dan nama pengguna/kata sandi) melalui objek.

`ClientConfiguration`

Konfigurasi Transportasi HTTP

Anda dapat mengkonfigurasi beberapa opsi transportasi HTTP dengan menggunakan [ClientConfiguration](#) objek. Opsi baru sesekali ditambahkan; untuk melihat daftar lengkap opsi yang dapat Anda ambil atau atur, lihat Referensi AWS SDK for Java API.

Note

Masing-masing nilai yang dapat dikonfigurasi memiliki nilai default yang ditentukan oleh konstanta. Untuk daftar nilai konstanta `ClientConfiguration`, lihat [Nilai Bidang Konstan](#) di Referensi AWS SDK for Java API.

Koneksi Maksimum

Anda dapat mengatur jumlah maksimum koneksi HTTP terbuka yang diizinkan dengan menggunakan file [ClientConfiguration.setMaxConnections](#) metode.

Important

Atur koneksi maksimum ke jumlah transaksi bersamaan untuk menghindari perselisihan koneksi dan kinerja yang buruk. Untuk nilai koneksi maksimum default, lihat [Nilai Bidang Konstan](#) di Referensi AWS SDK for Java API.

Timeout dan Penanganan Kesalahan

Anda dapat mengatur opsi yang terkait dengan batas waktu dan menangani kesalahan dengan koneksi HTTP.

- Batas Waktu Koneksi

Batas waktu koneksi adalah jumlah waktu (dalam milidetik) bahwa koneksi HTTP akan menunggu untuk membuat koneksi sebelum menyerah. Defaultnya adalah 10.000 ms.

Untuk menetapkan nilai ini sendiri, gunakan [ClientConfiguration.setConnectionTimeout](#) metode.

- Waktu Koneksi untuk Hidup (TTL)

Secara default, SDK akan mencoba menggunakan kembali koneksi HTTP selama mungkin. Dalam situasi kegagalan di mana koneksi dibuat ke server yang telah dibawa keluar dari layanan, memiliki TTL yang terbatas dapat membantu pemulihan aplikasi. Misalnya, pengaturan TTL 15 menit akan memastikan bahwa meskipun Anda memiliki koneksi yang dibuat ke server yang mengalami masalah, Anda akan membangun kembali koneksi ke server baru dalam waktu 15 menit.

Untuk mengatur TTL koneksi HTTP, gunakan [ClientConfigurationmetode.setConnectionTtl](#).

- **Mencoba Ulang Kesalahan Maksimum**

Jumlah percobaan ulang maksimum default untuk kesalahan yang dapat diambil adalah 3. Anda dapat menetapkan nilai yang berbeda dengan menggunakan [ClientConfiguration.setMaxErrorMetode](#) [coba lagi](#).

Alamat Lokal

[Untuk mengatur alamat lokal yang akan diikat oleh klien HTTP, gunakan ClientConfiguration.setLocalAddress.](#)

Petunjuk Ukuran Penyangga Soket TCP

Pengguna tingkat lanjut yang ingin menyetel parameter TCP tingkat rendah juga dapat mengatur petunjuk ukuran buffer TCP melalui objek [ClientConfiguration](#). Mayoritas pengguna tidak akan pernah perlu mengubah nilai-nilai ini, tetapi mereka disediakan untuk pengguna tingkat lanjut.

Ukuran buffer TCP yang optimal untuk suatu aplikasi sangat bergantung pada konfigurasi dan kemampuan jaringan dan sistem operasi. Misalnya, sebagian besar sistem operasi modern menyediakan logika penyetelan otomatis untuk ukuran buffer TCP. Ini dapat berdampak besar pada kinerja koneksi TCP yang dibuka cukup lama untuk penyetelan otomatis untuk mengoptimalkan ukuran buffer.

Ukuran buffer yang besar (misalnya, 2 MB) memungkinkan sistem operasi untuk menyangga lebih banyak data dalam memori tanpa memerlukan server jarak jauh untuk mengakui penerimaan informasi tersebut, sehingga dapat sangat berguna ketika jaringan memiliki latensi tinggi.

Ini hanya petunjuk, dan sistem operasi mungkin tidak menghormatinya. Saat menggunakan opsi ini, pengguna harus selalu memeriksa batas dan default yang dikonfigurasi sistem operasi. Sebagian besar sistem operasi memiliki batas ukuran buffer TCP maksimum yang dikonfigurasi, dan tidak akan membiarkan Anda melampaui batas itu kecuali Anda secara eksplisit menaikkan batas ukuran buffer TCP maksimum.

Banyak sumber daya tersedia untuk membantu mengonfigurasi ukuran buffer TCP dan pengaturan TCP khusus sistem operasi, termasuk yang berikut ini:

- [Penyetelan Tuan Rumah](#)

Kebijakan Kontrol Akses

AWS Kebijakan kontrol akses memungkinkan Anda menentukan kontrol akses berbutir halus pada sumber daya Anda. AWS Kebijakan kontrol akses terdiri dari kumpulan pernyataan, yang berbentuk:

Akun A memiliki izin untuk melakukan tindakan B pada sumber daya C di mana kondisi D berlaku.

Di mana:

- A adalah prinsipal - Akun AWS Yang membuat permintaan untuk mengakses atau memodifikasi salah satu AWS sumber daya Anda.
- B adalah tindakan - Cara di mana AWS sumber daya Anda sedang diakses atau dimodifikasi, seperti mengirim pesan ke Amazon SQS antrian, atau menyimpan objek dalam Amazon S3 ember.
- C adalah sumber daya - AWS Entitas yang prinsipal ingin mengakses, seperti Amazon SQS antrian, atau objek yang disimpan di Amazon S3.
- D adalah sekumpulan kondisi - Kendala opsional yang menentukan kapan harus mengizinkan atau menolak akses bagi prinsipal untuk mengakses sumber daya Anda. Banyak kondisi ekspresif tersedia, beberapa khusus untuk setiap layanan. Misalnya, Anda dapat menggunakan ketentuan tanggal untuk mengizinkan akses ke sumber daya Anda hanya setelah atau sebelum waktu tertentu.

Amazon S3 Contoh

Contoh berikut menunjukkan kebijakan yang memungkinkan siapa pun mengakses untuk membaca semua objek dalam bucket, tetapi membatasi akses untuk mengunggah objek ke bucket tersebut ke dua Akun AWS s tertentu (selain akun pemilik bucket).

```
Statement allowPublicReadStatement = new Statement(Effect.Allow)
    .withPrincipals(Principal.AllUsers)
    .withActions(S3Actions.GetObject)
    .withResources(new S3ObjectResource(myBucketName, "*"));
Statement allowRestrictedWriteStatement = new Statement(Effect.Allow)
    .withPrincipals(new Principal("123456789"), new Principal("876543210"))
    .withActions(S3Actions.PutObject)
    .withResources(new S3ObjectResource(myBucketName, "*"));

Policy policy = new Policy()
    .withStatements(allowPublicReadStatement, allowRestrictedWriteStatement);
```

```
AmazonS3 s3 = AmazonS3ClientBuilder.defaultClient();
s3.setBucketPolicy(myBucketName, policy.toJson());
```

Amazon SQS Contoh

Salah satu penggunaan kebijakan yang umum adalah untuk mengotorisasi Amazon SQS antrian untuk menerima pesan dari topik Amazon SNS.

```
Policy policy = new Policy().withStatements(
    new Statement(Effect.Allow)
        .withPrincipals(Principal.AllUsers)
        .withActions(SQSActions.SendMessage)
        .withConditions(ConditionFactory.newSourceArnCondition(myTopicArn)));

Map queueAttributes = new HashMap();
queueAttributes.put(QueueAttributeName.Policy.toString(), policy.toJson());

AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
sqs.setQueueAttributes(new SetQueueAttributesRequest(myQueueUrl, queueAttributes));
```

Contoh Amazon SNS

Beberapa layanan menawarkan ketentuan tambahan yang dapat digunakan dalam kebijakan. Amazon SNS menyediakan ketentuan untuk mengizinkan atau menolak langganan ke topik SNS berdasarkan protokol (misalnya, email, HTTP, HTTPS, Amazon SQS) dan titik akhir (misalnya, alamat email, URL, Amazon SQS ARN) dari permintaan untuk berlangganan topik.

```
Condition endpointCondition =
    SNSConditionFactory.newEndpointCondition("*@mycompany.com");

Policy policy = new Policy().withStatements(
    new Statement(Effect.Allow)
        .withPrincipals(Principal.AllUsers)
        .withActions(SNSActions.Subscribe)
        .withConditions(endpointCondition));

AmazonSNS sns = AmazonSNSClientBuilder.defaultClient();
sns.setTopicAttributes(
    new SetTopicAttributesRequest(myTopicArn, "Policy", policy.toJson()));
```


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 5 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.

Cara mengatur JVM TTL

Untuk memodifikasi TTL JVM, atur nilai properti keamanan [networkaddress.cache.ttl](#), atur `networkaddress.cache.ttl` properti dalam file untuk Java 8 atau file untuk Java 11 atau lebih tinggi. `$JAVA_HOME/jre/lib/security/java.security` `$JAVA_HOME/conf/security/java.security`

Berikut ini adalah cuplikan dari `java.security` file yang menunjukkan cache TTL diatur ke 5 detik.

```
#
# This is the "master security properties file".
#
# An alternate java.security properties file may be specified
...
# The Java-level namelookup cache policy for successful lookups:
#
# any negative value: caching forever
# any positive value: the number of seconds to cache an address for
# zero: do not cache
...
networkaddress.cache.ttl=5
...
```

Semua aplikasi yang berjalan pada JVM diwakili oleh variabel `$JAVA_HOME` lingkungan menggunakan pengaturan ini.

Mengaktifkan Metrik untuk AWS SDK for Java

AWS SDK for Java Dapat menghasilkan metrik untuk visualisasi dan pemantauan dengan [Amazon CloudWatch yang mengukur](#):

- kinerja aplikasi Anda saat mengakses AWS
- kinerja JVM Anda saat digunakan dengan AWS
- Rincian lingkungan runtime seperti memori heap, jumlah thread, dan deskriptor file yang dibuka

Cara Mengaktifkan Generasi Metrik SDK Java

Anda perlu menambahkan dependensi Maven berikut untuk mengaktifkan SDK untuk mengirim metrik. CloudWatch

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.12.490* </version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-cloudwatchmetrics</artifactId>
    <scope>provided</scope>
  </dependency>
  <!-- Other SDK dependencies. -->
</dependencies>
```

* Ganti nomor versi dengan versi terbaru SDK yang tersedia di [Maven Central](#).

AWS SDK for Java metrik dinonaktifkan secara default. Untuk mengaktifkannya untuk lingkungan pengembangan lokal Anda, sertakan properti sistem yang menunjuk ke file kredensi AWS keamanan Anda saat memulai JVM. Sebagai contoh:

```
-Dcom.amazonaws.sdk.enableDefaultMetrics=credentialFile=/path/aws.properties
```

Anda perlu menentukan jalur ke file kredensial Anda sehingga SDK dapat mengunggah titik data yang dikumpulkan untuk analisis selanjutnya. CloudWatch

Note

Jika Anda mengakses AWS dari sebuah Amazon EC2 instance menggunakan layanan metadata Amazon EC2 instance, Anda tidak perlu menentukan file kredensial. Dalam hal ini, Anda hanya perlu menentukan:

```
-Dcom.amazonaws.sdk.enableDefaultMetrics
```

Semua metrik yang ditangkap oleh AWS SDK for Java berada di bawah namespace AWSSDK/Java, dan diunggah ke CloudWatch wilayah default (us-east-1). Untuk mengubah wilayah, tentukan dengan menggunakan `cloudwatchRegion` atribut di properti sistem. Misalnya, untuk menyetel CloudWatch wilayah ke us-east-1, gunakan:

```
-Dcom.amazonaws.sdk.enableDefaultMetrics=credentialFile=/path/  
aws.properties,cloudwatchRegion={region_api_default}
```

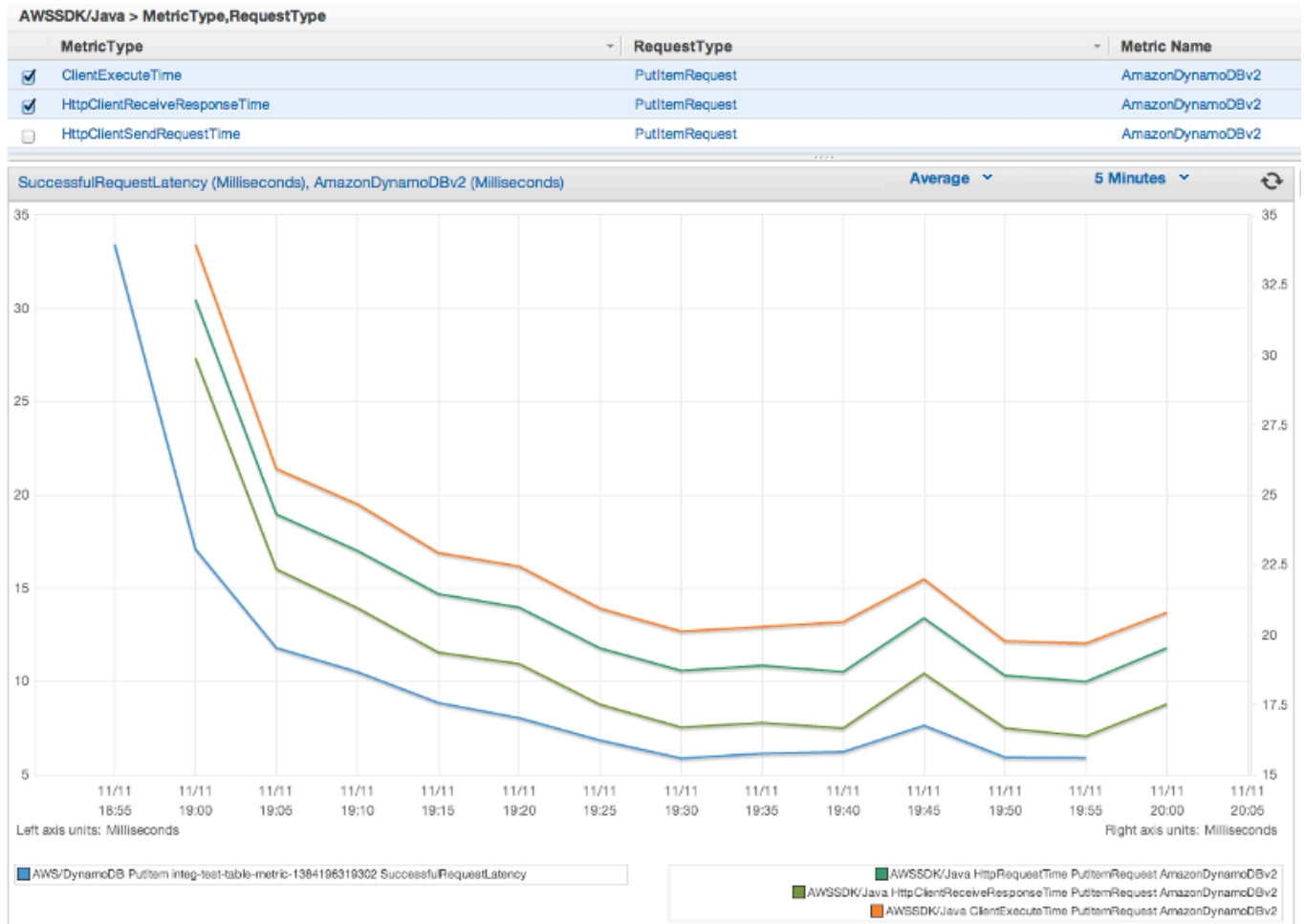
Setelah Anda mengaktifkan fitur, setiap kali ada permintaan layanan AWS dari, titik data metrik akan dibuat AWS SDK for Java, antri untuk ringkasan statistik, dan diunggah secara asinkron menjadi sekitar sekali setiap menit. CloudWatch Setelah metrik diunggah, Anda dapat memvisualisasikannya menggunakan [AWS Management Console](#) dan mengatur alarm pada potensi masalah seperti kebocoran memori, kebocoran deskriptor file, dan sebagainya.

Jenis Metrik yang Tersedia

Kumpulan metrik default dibagi menjadi tiga kategori utama:

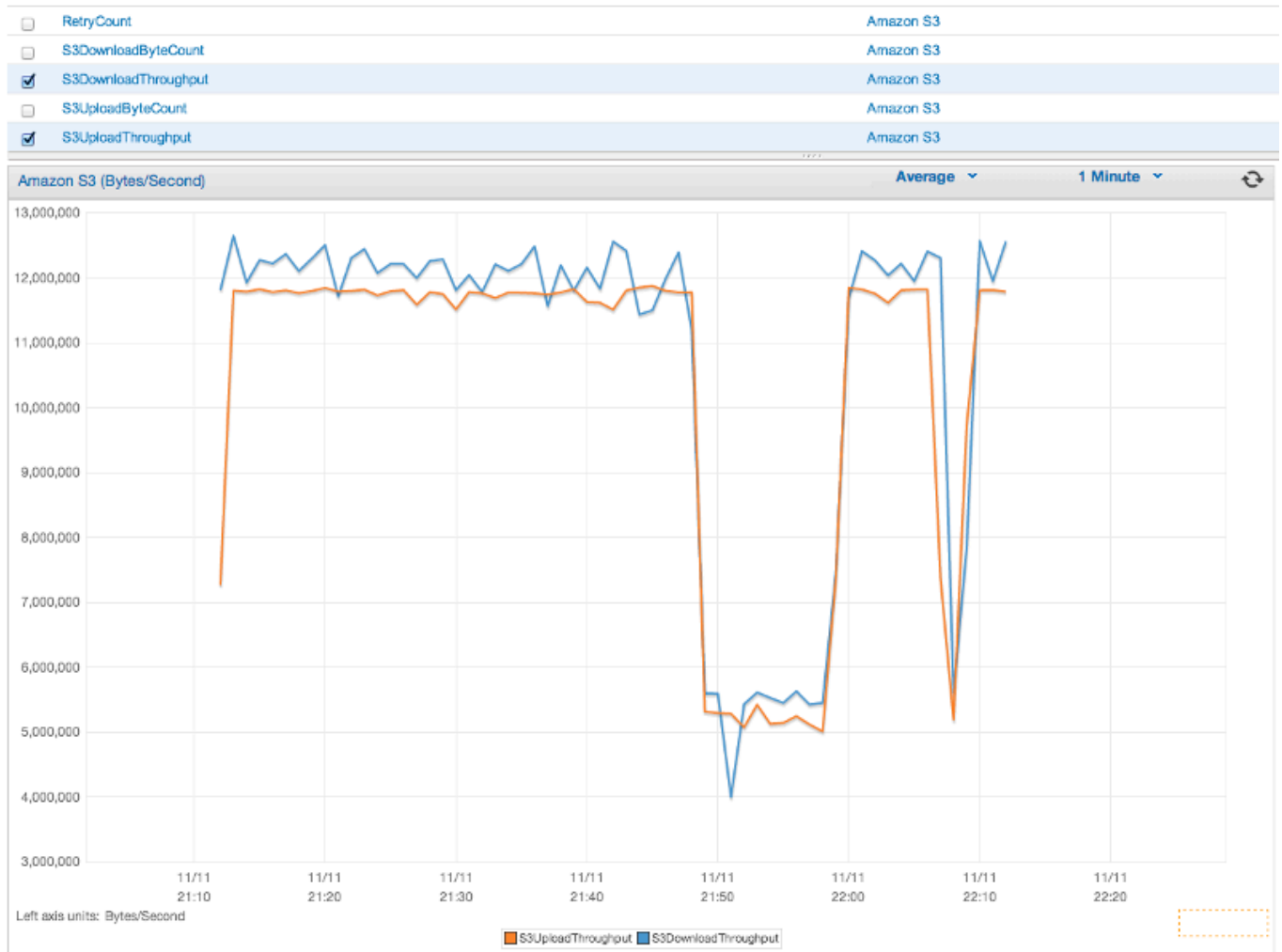
AWS Minta Metrik

- Mencakup area seperti latensi permintaan/respons HTTP, jumlah permintaan, pengecualian, dan percobaan ulang.



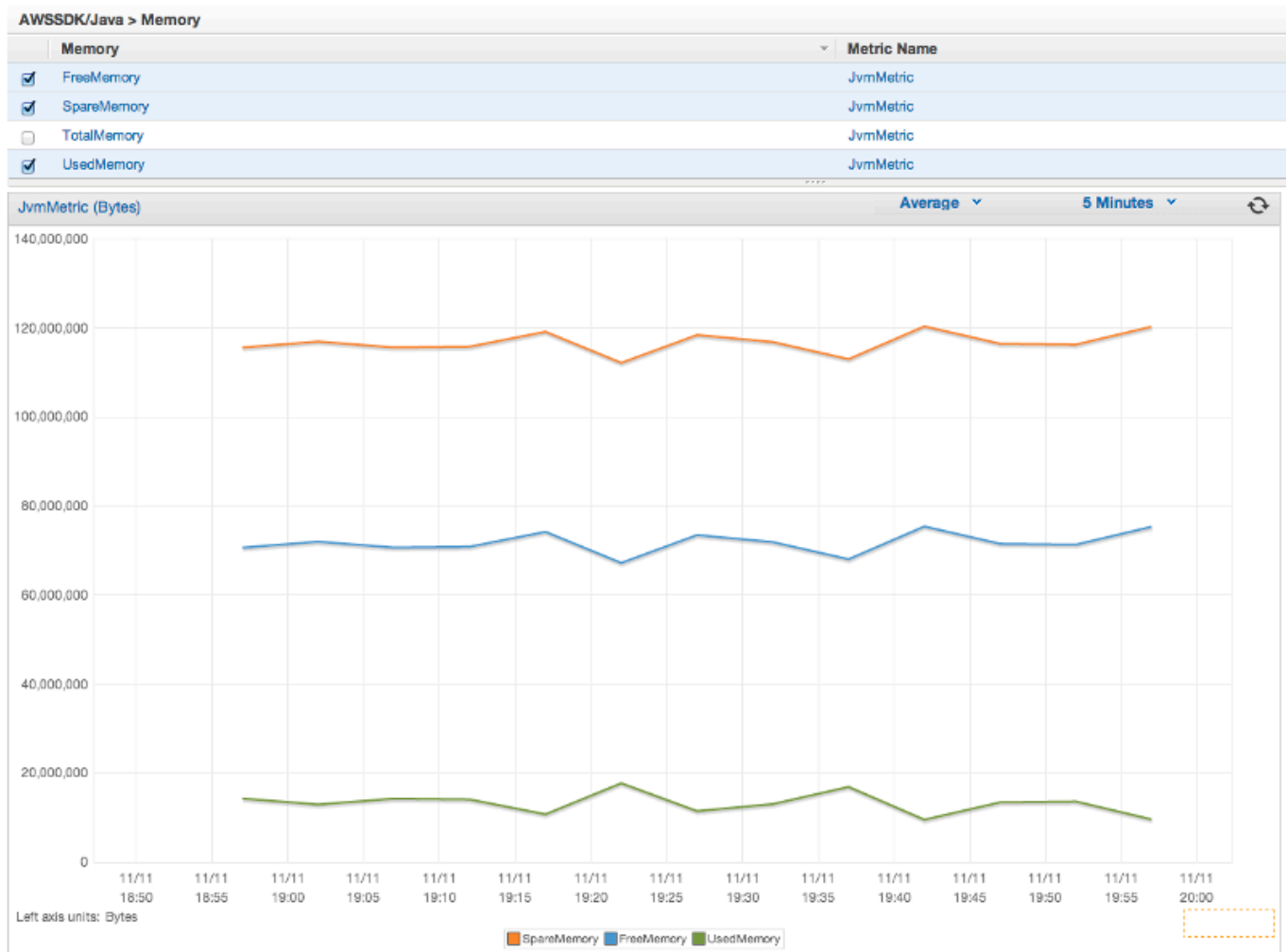
Layanan AWS Metrik

- Sertakan data Layanan AWS-spesifik, seperti throughput dan jumlah byte untuk unggahan dan unduhan S3.



Metrik Mesin

- Tutupi lingkungan runtime, termasuk memori heap, jumlah thread, dan deskriptor file terbuka.



Jika Anda ingin mengecualikan Metrik Mesin, tambahkan `excludeMachineMetrics` ke properti sistem:

```
-Dcom.amazonaws.sdk.enableDefaultMetrics=credentialFile=/path/
aws.properties,excludeMachineMetrics
```

Informasi Selengkapnya

- Lihat [ringkasan paket amazonaws/metrics](#) untuk daftar lengkap jenis metrik inti yang telah ditentukan sebelumnya.
- Pelajari tentang bekerja dengan CloudWatch menggunakan AWS SDK for Java dalam [CloudWatch Contoh Menggunakan AWS SDK for Java](#).

- Pelajari lebih lanjut tentang penyetelan kinerja dalam [Menyetel AWS SDK for Java untuk Meningkatkan Ketahanan posting blog](#).

AWS SDK for JavaContoh Kode

Bagian ini menyediakan tutorial dan contoh menggunakanAWS SDK for Java v1 untukAWS layanan program.

Menemukan kode sumber untuk contoh-contoh ini dan lain-lain dalamAWS dokumentasi [contoh kode repositori padaGitHub](#).

Untuk mengusulkan contoh kode baru bagi timAWS dokumentasi untuk mempertimbangkan memproduksi, buat permintaan baru. Tim ingin menghasilkan contoh kode yang mencakup skenario dan kasus penggunaan yang lebih luas, dibandingkan cuplikan kode sederhana yang hanya mencakup panggilan API individual. Untuk petunjuk, lihat [pedoman Berkontribusi](#) dalam contoh kode repository onGitHub..

AWS SDK for Java2.x

Pada tahun 2018,AWS merilis [AWS SDK for Java 2.x](#). Panduan ini berisi petunjuk tentang penggunaan Java SDK terbaru bersama dengan kode contoh.

Note

Lihat [Dokumentasi dan Sumber Daya Tambahan](#) untuk lebih banyak contoh dan sumber daya tambahan yang tersedia untukAWS SDK for Java pengembang!

Contoh CloudWatch MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrograman[CloudWatch](#)menggunakan[AWS SDK for Java](#).

Amazon CloudWatch memantauAmazon Web Services(AWS) sumber daya dan aplikasi yang Anda jalankan diAWSsecara waktu nyata. Anda dapat menggunakan CloudWatch untuk mengumpulkan dan menelusuri metrik, yang merupakan variabel yang dapat diukur untuk sumber daya dan aplikasi Anda. CloudWatch Alarm mengirimkan pemberitahuan atau secara otomatis melakukan perubahan pada sumber daya yang sedang dipantau berdasarkan aturan yang ditetapkan.

Untuk informasi selengkapnya tentang CloudWatch, lihat [Amazon CloudWatchPanduan Pengguna](#) .

Note

Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parameter [kode contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk membangun dan menjalankan.

Topik

- [Mendapatkan Metrik dari CloudWatch](#)
- [Memublikasikan Data Metrik Khusus](#)
- [Bekerja dengan CloudWatch Alarm](#)
- [Mengggunakan Tindakan Alarm di CloudWatch](#)
- [Mengirim Peristiwa keCloudWatch](#)

Mendapatkan Metrik dari CloudWatch

Metrik Listing

Daftar CloudWatch metrik, membuat [ListMetricsRequest](#) dan memanggil `AmazonCloudWatchClient.listMetrics` metode. Anda dapat menggunakan `ListMetricsRequest` untuk memfilter metrik yang dikembalikan dengan ruangnama, nama metrik, atau dimensi.

Note

Daftar metrik dan dimensi yang diposting oleh AWS layanan dapat ditemukan dalam <https://docs.aws.amazon.com/amazoncloudwatch/latest/monitoring-cw-support-for-AWS.html> [Amazon CloudWatch Referensi Metrik dan Dimensi] di Amazon CloudWatch Panduan Pengguna.

IMPOR

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
```

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.ListMetricsRequest;
import com.amazonaws.services.cloudwatch.model.ListMetricsResult;
import com.amazonaws.services.cloudwatch.model.Metric;
```

Kode

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

ListMetricsRequest request = new ListMetricsRequest()
    .withMetricName(name)
    .withNamespace(namespace);

boolean done = false;

while(!done) {
    ListMetricsResult response = cw.listMetrics(request);

    for(Metric metric : response.getMetrics()) {
        System.out.printf(
            "Retrieved metric %s", metric.getMetricName());
    }

    request.setNextToken(response.getNextToken());

    if(response.getNextToken() == null) {
        done = true;
    }
}
```

Metrik dikembalikan dalam [ListMetricsResult](#) dengan memanggil `getMetrics` metode. Hasilnya mungkin `paged`. Untuk mengambil batch berikutnya hasil, hubungi `setNextToken` pada objek permintaan asli dengan nilai kembali dari `ListMetricsResult` objek `getNextToken` metode, dan lulus objek permintaan dimodifikasi kembali ke panggilan lain untuk `listMetrics`.

Informasi Selengkapnya

- [ListMetrics](#) di Amazon CloudWatch Referensi API.

Memublikasikan Data Metrik Khusus

Sejumlah AWS memublikasikan layanan [Metrik mereka sendiri](#) di ruang nama yang dimulai dengan "AWS". Anda juga dapat memublikasikan data metrik kustom menggunakan namespace Anda sendiri (asalkan tidak dimulai dengan "AWS").

Publikasikan Data Metrik Kustom

Untuk memublikasikan data metrik Anda sendiri, hubungi `AmazonCloudWatchClient'sputMetricData` metode dengan [PutMetricDataRequest](#). Parameter `PutMetricDataRequest` harus menyertakan namespace khusus untuk digunakan untuk data, dan informasi tentang titik data itu sendiri dalam [MetricDatum](#) objek.

Note

Anda tidak dapat menentukan namespace yang dimulai dengan "AWS". Ruang nama yang dimulai dengan "AWS" disimpan untuk digunakan oleh Amazon Web Services produk.

Impor

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.Dimension;
import com.amazonaws.services.cloudwatch.model.MetricDatum;
import com.amazonaws.services.cloudwatch.model.PutMetricDataRequest;
import com.amazonaws.services.cloudwatch.model.PutMetricDataResult;
import com.amazonaws.services.cloudwatch.model.StandardUnit;
```

Kode

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

Dimension dimension = new Dimension()
    .withName("UNIQUE_PAGES")
    .withValue("URLS");

MetricDatum datum = new MetricDatum()
    .withMetricName("PAGES_VISITED")
    .withUnit(StandardUnit.None)
```

```
.withValue(data_point)
.withDimensions(dimension);

PutMetricDataRequest request = new PutMetricDataRequest()
    .withNamespace("SITE/TRAFFIC")
    .withMetricData(datum);

PutMetricDataResult response = cw.putMetricData(request);
```

Informasi Selengkapnya

- [Menggunakan Amazon CloudWatch Metrik](#) di dalam Amazon CloudWatch Panduan Pengguna.
- [AWS Namespace](#) di dalam Amazon CloudWatch Panduan Pengguna.
- [PutMetricData](#) di dalam Amazon CloudWatch Referensi API.

Bekerja dengan CloudWatch Alarm

Buat alarm

Untuk membuat alarm didasarkan pada CloudWatch metrik, panggil `AmazonCloudWatchClient.putMetricAlarm` metode dengan [PutMetricAlarmRequest](#) diisi dengan kondisi alarm.

Impor

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.ComparisonOperator;
import com.amazonaws.services.cloudwatch.model.Dimension;
import com.amazonaws.services.cloudwatch.model.PutMetricAlarmRequest;
import com.amazonaws.services.cloudwatch.model.PutMetricAlarmResult;
import com.amazonaws.services.cloudwatch.model.StandardUnit;
import com.amazonaws.services.cloudwatch.model.Statistic;
```

Kode

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

Dimension dimension = new Dimension()
```

```
.withName("InstanceId")
.withValue(instanceId);

PutMetricAlarmRequest request = new PutMetricAlarmRequest()
    .withAlarmName(alarmName)
    .withComparisonOperator(
        ComparisonOperator.GreaterThanThreshold)
    .withEvaluationPeriods(1)
    .withMetricName("CPUUtilization")
    .withNamespace("{AWS}/EC2")
    .withPeriod(60)
    .withStatistic(Statistic.Average)
    .withThreshold(70.0)
    .withActionsEnabled(false)
    .withAlarmDescription(
        "Alarm when server CPU utilization exceeds 70%")
    .withUnit(StandardUnit.Seconds)
    .withDimensions(dimension);

PutMetricAlarmResult response = cw.putMetricAlarm(request);
```

Daftar alarm

Untuk mencantumkan CloudWatch alarm yang telah Anda buat, panggil `AmazonCloudWatchClient`'s `describeAlarms` Metode dengan [DescribeAlarmsRequest](#) yang dapat Anda gunakan untuk mengatur opsi untuk hasilnya.

Impor

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.DescribeAlarmsRequest;
import com.amazonaws.services.cloudwatch.model.DescribeAlarmsResult;
import com.amazonaws.services.cloudwatch.model.MetricAlarm;
```

Kode

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

boolean done = false;
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
```

```
while(!done) {  
  
    DescribeAlarmsResult response = cw.describeAlarms(request);  
  
    for(MetricAlarm alarm : response.getMetricAlarms()) {  
        System.out.printf("Retrieved alarm %s", alarm.getAlarmName());  
    }  
  
    request.setNextToken(response.getNextToken());  
  
    if(response.getNextToken() == null) {  
        done = true;  
    }  
}
```

Daftar alarm dapat diperoleh dengan menelepon `getMetricAlarms` pada [DescribeAlarmsResult](#) yang dikembalikan oleh `describeAlarms`.

Hasilnya mungkin `paged`. Untuk mengambil batch berikutnya hasil, hubungi `setNextToken` pada objek permintaan asli dengan nilai kembali dari `DescribeAlarmsResult` objek `getNextToken` metode, dan lulus objek permintaan dimodifikasi kembali ke panggilan lain untuk `describeAlarms`.

Note

Anda juga dapat mengambil alarm untuk metrik tertentu dengan menggunakan `AmazonCloudWatchClient`'s `describeAlarmsForMetric` metode. Penggunaannya mirip dengan `describeAlarms`.

Hapus Alarm

Untuk menghapus CloudWatch alarm, panggil `AmazonCloudWatchClient` `deleteAlarms` metode dengan [DeleteAlarmsRequest](#) berisi satu atau lebih nama alarm yang ingin Anda hapus.

Impor

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;  
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;  
import com.amazonaws.services.cloudwatch.model.DeleteAlarmsRequest;
```

```
import com.amazonaws.services.cloudwatch.model.DeleteAlarmsResult;
```

Kode

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

DeleteAlarmsRequest request = new DeleteAlarmsRequest()
    .withAlarmNames(alarm_name);

DeleteAlarmsResult response = cw.deleteAlarms(request);
```

Informasi Selengkapnya

- [Membuat Amazon CloudWatch Alarm](#) di Amazon CloudWatch Panduan Pengguna
- [PutMetricAlarm](#) di Amazon CloudWatch Referensi API
- [DescribeAlarms](#) di Amazon CloudWatch Referensi API
- [DeleteAlarms](#) di Amazon CloudWatch Referensi API

Menggunakan Tindakan Alarm di CloudWatch

Menggunakan CloudWatch Tindakan alarm, Anda dapat membuat alarm yang melakukan tindakan seperti secara otomatis menghentikan, mengakhiri, menyalakan ulang Amazon EC2 contoh.

Note

Tindakan alarm dapat ditambahkan ke alarm dengan menggunakan [PutMetricAlarmRequest](#)'s `setAlarmActions` Metode ketika [membuat alarm](#).

Mengaktifkan Tindakan Alarm

Untuk mengaktifkan tindakan alarm untuk CloudWatch alarm, panggil `AmazonCloudWatchClient.enableAlarmActions` dengan [EnableAlarmActionsRequest](#) berisi satu atau beberapa nama alarm yang tindakannya ingin Anda aktifkan.

Impor

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
```

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.EnableAlarmActionsRequest;
import com.amazonaws.services.cloudwatch.model.EnableAlarmActionsResult;
```

Kode

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

EnableAlarmActionsRequest request = new EnableAlarmActionsRequest()
    .withAlarmNames(alarm);

EnableAlarmActionsResult response = cw.enableAlarmActions(request);
```

Tindakan Alarm

Untuk menonaktifkan tindakan alarm untuk CloudWatch alarm, panggil `AmazonCloudWatchClient.disableAlarmActions` dengan [DisableAlarmActionsRequest](#) berisi satu atau lebih nama alarm yang tindakannya ingin Anda nonaktifkan.

Impor

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.DisableAlarmActionsRequest;
import com.amazonaws.services.cloudwatch.model.DisableAlarmActionsResult;
```

Kode

```
final AmazonCloudWatch cw =
    AmazonCloudWatchClientBuilder.defaultClient();

DisableAlarmActionsRequest request = new DisableAlarmActionsRequest()
    .withAlarmNames(alarmName);

DisableAlarmActionsResult response = cw.disableAlarmActions(request);
```

Informasi Selengkapnya

- [Buat Alarm untuk Menghentikan, Mengakhiri, Menyalakan Ulang, atau Memulihkan Contoh](#) di dalam [Amazon CloudWatch Panduan Pengguna](#)

- [PutMetricAlarm](#) di dalam Amazon CloudWatch Referensi API
- [EnableAlarmActions](#) di dalam Amazon CloudWatch Referensi API
- [DisableAlarmActions](#) di dalam Amazon CloudWatch Referensi API

Mengirim Peristiwa ke CloudWatch

CloudWatch Peristiwa memberikan aliran kejadian sistem secara hampir waktunya yang menjelaskan perubahan dalam AWS sumber daya untuk Amazon EC2 contoh, Lambda fungsi, Kinesis aliran, Amazon EC2 tugas, Step Functions mesin status, Amazon SNS topik, Amazon SQS antrian, atau target bawaan. Anda dapat mencocokkan peristiwa dan merutkannya ke satu atau beberapa fungsi atau pengaliran target dengan menggunakan aturan sederhana.

Tambahkan Acara

Untuk menambahkan kustom CloudWatch peristiwa, memanggil `AmazonCloudWatchEventsClient.putEvents` dengan metode [PutEventsRequest](#) objek yang berisi satu atau beberapa [PutEventsRequestEntry](#) objek yang memberikan rincian tentang setiap acara. Anda dapat menentukan beberapa parameter untuk entri seperti sumber dan jenis acara, sumber daya yang terkait dengan acara, dan sebagainya.

Note

Anda dapat menentukan maksimal 10 peristiwa per panggilan ke `putEvents`.

Impor

```
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutEventsRequest;
import com.amazonaws.services.cloudwatchevents.model.PutEventsRequestEntry;
import com.amazonaws.services.cloudwatchevents.model.PutEventsResult;
```

Kode

```
final AmazonCloudWatchEvents cwe =
    AmazonCloudWatchEventsClientBuilder.defaultClient();
```

```
final String EVENT_DETAILS =
    "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

PutEventsRequestEntry request_entry = new PutEventsRequestEntry()
    .withDetail(EVENT_DETAILS)
    .withDetailType("sampleSubmitted")
    .withResources(resource_arn)
    .withSource("aws-sdk-java-cloudwatch-example");

PutEventsRequest request = new PutEventsRequest()
    .withEntries(request_entry);

PutEventsResult response = cwe.putEvents(request);
```

Tambahkan Aturan

Untuk membuat atau memperbarui aturan, panggil `AmazonCloudWatchEventsClient.putRule` dengan metode [PutRuleRequest](#) dengan nama aturan dan parameter opsional seperti [Pola peristiwa](#), IAM peran untuk mengasosiasikan dengan aturan, dan [Ekspresi penjadwalkan](#) yang menjelaskan seberapa sering aturan dijalankan.

Impor

```
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutRuleRequest;
import com.amazonaws.services.cloudwatchevents.model.PutRuleResult;
import com.amazonaws.services.cloudwatchevents.model.RuleState;
```

Kode

```
final AmazonCloudWatchEvents cwe =
    AmazonCloudWatchEventsClientBuilder.defaultClient();

PutRuleRequest request = new PutRuleRequest()
    .withName(rule_name)
    .withRoleArn(role_arn)
    .withScheduleExpression("rate(5 minutes)")
    .withState(RuleState.ENABLED);

PutRuleResult response = cwe.putRule(request);
```

Tambah Target

Target adalah sumber daya yang dipanggil ketika suatu aturan dipicu. Contoh target termasuk Amazon EC2, Lambda fungsi, Kinesis aliran, Amazon EC2 tugas, Step Functions mesin negara, dan target built-in.

Untuk menambahkan target ke aturan, panggil `AmazonCloudWatchEventsClient.putTargets` dengan metode [PutTargetsRequest](#) berisi aturan untuk memperbarui dan daftar target untuk ditambahkan ke aturan.

Impor

```
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutTargetsRequest;
import com.amazonaws.services.cloudwatchevents.model.PutTargetsResult;
import com.amazonaws.services.cloudwatchevents.model.Target;
```

Kode

```
final AmazonCloudWatchEvents cwe =
    AmazonCloudWatchEventsClientBuilder.defaultClient();

Target target = new Target()
    .withArn(function_arn)
    .withId(target_id);

PutTargetsRequest request = new PutTargetsRequest()
    .withTargets(target)
    .withRule(rule_name);

PutTargetsResult response = cwe.putTargets(request);
```

Informasi Selengkapnya

- [Menambahkan Peristiwa dengan PutEvents](#) di Amazon CloudWatch Events Panduan Pengguna
- [Eksresi Jadwal untuk Aturan](#) di Amazon CloudWatch Events Panduan Pengguna
- [Jenis Peristiwa untuk CloudWatch Peristiwa](#) di Amazon CloudWatch Events Panduan Pengguna
- [Pola Peristiwa dan Peristiwa](#) di Amazon CloudWatch Events Panduan Pengguna

- [PutEvents](#) di Amazon CloudWatch Events Referensi API
- [PutTargets](#) di Amazon CloudWatch Events Referensi API
- [PutRule](#) di Amazon CloudWatch Events Referensi API

DynamoDB Contoh Menggunakan AWS SDK for Java

Bagian ini menyediakan contoh pemrograman [DynamoDB](#) menggunakan [AWS SDK for Java](#).

Note

Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parameter [contoh kode lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk membangun dan menjalankan.

Topik

- [Cara dengan dengan DynamoDB](#)
- [Cara Menggunakan Item DynamoDB](#)

Cara dengan dengan DynamoDB

Tabel adalah wadah untuk semua item dalam DynamoDB database. Sebelum Anda dapat melakukan tambahkan atau hapus data dari DynamoDB, Anda harus membuat tabel.

Untuk setiap tabel, Anda harus menentukan:

- Nama tabel yang unik untuk akun dan wilayah Anda.
- Kunci utama yang setiap nilai harus unik; tidak ada dua item dalam tabel Anda dapat memiliki nilai kunci utama yang sama.

Kunci primer bisa sederhana, terdiri dari satu partisi (HASH) kunci, atau komposit, yang terdiri dari partisi dan semacam (RANGE) kunci.

Setiap nilai kunci memiliki tipe data yang terkait, disebutkan oleh [ScalarAttributeType](#) kelas. Nilai kunci dapat biner (B), numerik (N), atau string (S). Untuk informasi selengkapnya, lihat [Aturan Penamaan dan Jenis Data](#) di Panduan Amazon DynamoDB Pengembang.

- Nilai throughput yang disediakan yang menentukan jumlah unit kapasitas baca/tulis cadangan untuk tabel.

Note

[Amazon DynamoDB harga](#) didasarkan pada nilai throughput yang disediakan yang Anda tetapkan pada tabel Anda, jadi cadangan hanya kapasitas sebanyak yang Anda pikir Anda perlukan untuk meja Anda.

Throughput yang disediakan untuk tabel dapat dimodifikasi kapan saja, sehingga Anda dapat menyesuaikan kapasitas jika kebutuhan Anda berubah.

Buat Tabel

Gunakan `createTable` metode [DynamoDBKlien](#) 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. Hanya menentukan atribut tabel kunci saat membuat DynamoDB tabel Anda.

Note

Jika tabel dengan nama yang Anda pilih sudah ada, sebuah [AmazonServiceException](#) dilemparkan.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.CreateTableResult;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;
```

Buat Tabel dengan Kunci Utama Sederhana

Kode ini membuat tabel dengan kunci primer sederhana ("Nama").

Kode

```
CreateTableRequest request = new CreateTableRequest()
    .withAttributeDefinitions(new AttributeDefinition(
        "Name", ScalarAttributeType.S))
    .withKeySchema(new KeySchemaElement("Name", KeyType.HASH))
    .withProvisionedThroughput(new ProvisionedThroughput(
        new Long(10), new Long(10)))
    .withTableName(table_name);

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    CreateTableResult result = ddb.createTable(request);
    System.out.println(result.getTableDescription().getTableName());
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) GitHub.

Buat Tabel dengan Kunci Utama Komposit

Tambahkan yang lain [AttributeDefinition](#) dan [KeySchemaElement](#) ke [CreateTableRequest](#).

Kode

```
CreateTableRequest request = new CreateTableRequest()
    .withAttributeDefinitions(
        new AttributeDefinition("Language", ScalarAttributeType.S),
        new AttributeDefinition("Greeting", ScalarAttributeType.S))
    .withKeySchema(
        new KeySchemaElement("Language", KeyType.HASH),
        new KeySchemaElement("Greeting", KeyType.RANGE))
    .withProvisionedThroughput(
        new ProvisionedThroughput(new Long(10), new Long(10)))
    .withTableName(table_name);
```

Lihat [contoh lengkapnya](#) GitHub.

Daftar Tabel

Anda dapat mencantumkan tabel di wilayah tertentu dengan memanggil `listTables` metode [DynamoDBKlien](#).

Note

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, sebuah [ResourceNotFoundException](#) dilemparkan.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.ListTablesRequest;
import com.amazonaws.services.dynamodbv2.model.ListTablesResult;
```

Kode

```
final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

ListTablesRequest request;

boolean more_tables = true;
String last_name = null;

while(more_tables) {
    try {
        if (last_name == null) {
            request = new ListTablesRequest().withLimit(10);
        }
        else {
            request = new ListTablesRequest()
                .withLimit(10)
                .withExclusiveStartTableName(last_name);
        }

        ListTablesResult table_list = ddb.listTables(request);
        List<String> table_names = table_list.getTableNames();
    }
}
```

```
if (table_names.size() > 0) {
    for (String cur_name : table_names) {
        System.out.format("* %s\n", cur_name);
    }
} else {
    System.out.println("No tables found!");
    System.exit(0);
}

last_name = table_list.getLastEvaluatedTableName();
if (last_name == null) {
    more_tables = false;
}
```

Secara default, hingga 100 tabel dikembalikan per panggilan—gunakan `getLastEvaluatedTableName` pada [ListTablesResult](#) 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](#) GitHub.

Jelaskan (Dapatkan Informasi tentang) Tabel

Panggil `describeTable` metode [DynamoDBKlien](#).

Note

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, sebuah [ResourceNotFoundException](#) dilemparkan.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughputDescription;
import com.amazonaws.services.dynamodbv2.model.TableDescription;
```

Kode


```
final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    TableDescription table_info =
        ddb.describeTable(table_name).getTable();

    if (table_info != null) {
        System.out.format("Table name   : %s\n",
            table_info.getTable_name());
        System.out.format("Table ARN   : %s\n",
            table_info.getTableArn());
        System.out.format("Status      : %s\n",
            table_info.getTableStatus());
        System.out.format("Item count  : %d\n",
            table_info.getItemCount().longValue());
        System.out.format("Size (bytes): %d\n",
            table_info.getTableSizeBytes().longValue());

        ProvisionedThroughputDescription throughput_info =
            table_info.getProvisionedThroughput();
        System.out.println("Throughput");
        System.out.format("  Read Capacity : %d\n",
            throughput_info.getReadCapacityUnits().longValue());
        System.out.format("  Write Capacity: %d\n",
            throughput_info.getWriteCapacityUnits().longValue());

        List<AttributeDefinition> attributes =
            table_info.getAttributeDefinitions();
        System.out.println("Attributes");
        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n",
                a.getAttributeName(), a.getAttributeType());
        }
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) GitHub.

Memodifikasi (Update) Tabel

Anda dapat memodifikasi nilai throughput yang disediakan tabel Anda kapan saja dengan memanggil `updateTable` metode [DynamoDBKlien](#).

Note

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, sebuah [ResourceNotFoundException](#) dilemparkan.

Impor

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.AmazonServiceException;
```

Kode

```
ProvisionedThroughput table_throughput = new ProvisionedThroughput(
    read_capacity, write_capacity);

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    ddb.updateTable(table_name, table_throughput);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) GitHub.

Menghapus Tabel

Panggil `deleteTable` metode [DynamoDBKlien](#) dan berikan nama tabel.

Note

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, sebuah [ResourceNotFoundException](#) dilemparkan.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
```

Kode

```
final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    ddb.deleteTable(table_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) GitHub.

Info Selengkapnya

- [Pedoman untuk Bekerja dengan Tabel](#) di Panduan Amazon DynamoDB Pengembang
- [Bekerja dengan Tabel DynamoDB di](#) dalam Panduan Amazon DynamoDB Pengembang

Cara Menggunakan ItemDynamoDB

Masuk DynamoDB, item adalah koleksi atribut, yang masing-masing memiliki nama dan nilai. Nilai atribut dapat berupa skalar, set, atau jenis dokumen. Untuk informasi selengkapnya, lihat [Peraturan Penamaan dan Jenis Data](#) di Amazon DynamoDB Panduan Developer.

Ambil (Dapatkan) Item dari Tabel

Panggil `AmazonDynamoDB.getItem` dan menyebarkan `GetItemRequest` objek dengan nama tabel dan nilai kunci primer dari item yang Anda inginkan. Ia mengembalikan `GetItemResult` objek.

Anda dapat menggunakan kembali `getItemResult` objek `getItem()` metode untuk mengambil `Petaku` kunci (String) dan nilai (`AttributeValue`) pasangan yang terkait dengan item.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
```

Kode

```
HashMap<String, AttributeValue> key_to_get =
    new HashMap<String, AttributeValue>();

key_to_get.put("DATABASE_NAME", new AttributeValue(name));

GetItemRequest request = null;
if (projection_expression != null) {
    request = new GetItemRequest()
        .withKey(key_to_get)
        .withTableName(table_name)
        .withProjectionExpression(projection_expression);
} else {
    request = new GetItemRequest()
        .withKey(key_to_get)
        .withTableName(table_name);
}

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    Map<String, AttributeValue> returned_item =
        ddb.getItem(request).getItem();
    if (returned_item != null) {
        Set<String> keys = returned_item.keySet();
        for (String key : keys) {
            System.out.format("%s: %s\n",
                key, returned_item.get(key).toString());
        }
    }
}
```

```
    } else {  
        System.out.format("No item found with the key %s!\n", name);  
    }  
} catch (AmazonServiceException e) {  
    System.err.println(e.getMessage());  
    System.exit(1);  
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Tambahkan Item Baru ke Tabel

Buat [Pet](#)apasan nilai-kunci yang mewakili atribut item. Ini harus mencakup nilai-nilai untuk bidang kunci utama tabel. Jika item yang diidentifikasi oleh kunci primer sudah ada, bidangnyadi diperbarui oleh permintaan.

Note

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, a [Pengecualian Sumber Daya Tidak Ditemukan](#) dilemparkan.

Impor

```
import com.amazonaws.AmazonServiceException;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.model.AttributeValue;  
import com.amazonaws.services.dynamodbv2.model.ResourceNotFoundException;  
import java.util.ArrayList;
```

Kode

```
HashMap<String, AttributeValue> item_values =  
    new HashMap<String, AttributeValue>();  
  
item_values.put("Name", new AttributeValue(name));  
  
for (String[] field : extra_fields) {  
    item_values.put(field[0], new AttributeValue(field[1]));  
}  
  
final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();
```

```
try {
    ddb.putItem(table_name, item_values);
} catch (ResourceNotFoundException e) {
    System.err.format("Error: The table \"%s\" can't be found.\n", table_name);
    System.err.println("Be sure that it exists and that you've typed its name
correctly!");
    System.exit(1);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Memperbarui Item yang Ada dalam Tabel

Anda dapat memperbarui atribut untuk item yang sudah ada dalam tabel dengan menggunakan `AmazonDynamoDB.updateItem` metode, memberikan nama tabel, nilai kunci utama, dan peta bidang untuk memperbarui.

Note

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, atau jika item yang diidentifikasi oleh kunci utama yang Anda lewati tidak ada, [aPengecualianSumberDayaTidakDitemukan](#) dilemparkan.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeAction;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.AttributeValueUpdate;
import com.amazonaws.services.dynamodbv2.model.ResourceNotFoundException;
import java.util.ArrayList;
```

Kode

```
HashMap<String,AttributeValue> item_key =
    new HashMap<String,AttributeValue>();
```

```
item_key.put("Name", new AttributeValue(name));

HashMap<String,AttributeValueUpdate> updated_values =
    new HashMap<String,AttributeValueUpdate>();

for (String[] field : extra_fields) {
    updated_values.put(field[0], new AttributeValueUpdate(
        new AttributeValue(field[1]), AttributeAction.PUT));
}

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    ddb.updateItem(table_name, item_key, updated_values);
} catch (ResourceNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Menggunakan class DynamoDBMapper

Parameter [AWS SDK for Java](#) menyediakan [DynamoDBMapper](#) class, yang memungkinkan Anda memetakan kelas sisi klien Amazon DynamoDB tabel. Untuk menggunakan [DynamoDBMapper](#) kelas, Anda mendefinisikan hubungan antara item dalam DynamoDB tabel dan instans objek yang sesuai dalam kode Anda dengan menggunakan anotasi (seperti yang ditunjukkan pada contoh kode berikut). Parameter [DynamoDBMapper](#) class memungkinkan Anda mengakses tabel; melakukan berbagai operasi buat, baca, perbarui, dan hapus (CRUD) operasi; dan menjalankan kueri.

Note

Parameter [DynamoDBMapper](#) kelas tidak memungkinkan Anda membuat, memperbarui, atau menghapus tabel.

Impor

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
```

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
import com.amazonaws.services.dynamodbv2.model.AmazonDynamoDBException;
```

Kode

Contoh kode Java berikut menunjukkan kepada Anda cara menambahkan konten keMusiktabel dengan menggunakan[DynamoDBMapper](#)kelas. Setelah konten ditambahkan ke tabel, perhatikan bahwa item dimuat dengan menggunakanPartisidanUrutkankunci. makaPenghargaanitem diperbarui. Untuk informasi tentang membuatMusikmeja, lihat[Buat Tabel](#)diAmazon DynamoDBPanduan Developer.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
MusicItems items = new MusicItems();

try{
    // Add new content to the Music table
    items.setArtist(artist);
    items.setSongTitle(songTitle);
    items.setAlbumTitle(albumTitle);
    items.setAwards(Integer.parseInt(awards)); //convert to an int

    // Save the item
    DynamoDBMapper mapper = new DynamoDBMapper(client);
    mapper.save(items);

    // Load an item based on the Partition Key and Sort Key
    // Both values need to be passed to the mapper.load method
    String artistName = artist;
    String songQueryTitle = songTitle;

    // Retrieve the item
    MusicItems itemRetrieved = mapper.load(MusicItems.class, artistName,
songQueryTitle);
    System.out.println("Item retrieved:");
    System.out.println(itemRetrieved);

    // Modify the Award value
    itemRetrieved.setAwards(2);
```



```
        mapper.save(itemRetrieved);
        System.out.println("Item updated:");
        System.out.println(itemRetrieved);

        System.out.print("Done");
    } catch (AmazonDynamoDBException e) {
        e.printStackTrace();
    }
}

@DynamoDBTable(tableName="Music")
public static class MusicItems {

    //Set up Data Members that correspond to columns in the Music table
    private String artist;
    private String songTitle;
    private String albumTitle;
    private int awards;

    @DynamoDBHashKey(attributeName="Artist")
    public String getArtist() {
        return this.artist;
    }

    public void setArtist(String artist) {
        this.artist = artist;
    }

    @DynamoDBRangeKey(attributeName="SongTitle")
    public String getSongTitle() {
        return this.songTitle;
    }

    public void setSongTitle(String title) {
        this.songTitle = title;
    }

    @DynamoDBAttribute(attributeName="AlbumTitle")
    public String getAlbumTitle() {
        return this.albumTitle;
    }

    public void setAlbumTitle(String title) {
        this.albumTitle = title;
    }
}
```

```
    }

    @DynamoDBAttribute(attributeName="Awards")
    public int getAwards() {
        return this.awards;
    }

    public void setAwards(int awards) {
        this.awards = awards;
    }
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Info Selengkapnya

- [Pedoman Cara Menggunakan Item](#) di Amazon DynamoDB Panduan Pengembang
- [Cara Menggunakan Item DynamoDB](#) di Amazon DynamoDB Panduan Pengembang

Amazon EC2 Contoh Menggunakan AWS SDK for Java

Bagian ini menyediakan contoh pemrograman [Amazon EC2](#) dengan AWS SDK for Java.

Topik

- [Tutorial: Memulai instans EC2](#)
- [Menggunakan Peran IAM untuk Memberikan Akses ke AWS Sumber Daya di Amazon EC2](#)
- [Tutorial: Amazon EC2 Instans Spot](#)
- [Tutorial: Lanjutan Amazon EC2 Manajemen permintaan spot](#)
- [Mengelola Amazon EC2 Instans](#)
- [Menggunakan Alamat IP Elastis di Amazon EC2](#)
- [Gunakan wilayah dan availability zone](#)
- [Bekerja dengan Amazon EC2 Pasangan Kunci](#)
- [Bekerja dengan Grup Amazon EC2](#)

Tutorial: Memulai instans EC2

Tutorial ini menunjukkan bagaimana menggunakan AWS SDK for Java untuk memulai instans EC2.

Topik

- [Prasyarat](#)
- [Buat Grup Amazon EC2 Keamanan](#)
- [Membuat Pasangan Kunci](#)
- [Jalankan Amazon EC2 Instans](#)

Prasyarat

Sebelum memulai, pastikan bahwa Anda telah membuat Akun AWS dan bahwa Anda telah mengatur AWS kredensi. Untuk informasi selengkapnya, lihat [Memulai](#).

Buat Grup Amazon EC2 Keamanan

EC2-Classic akan pensiun

Warning

Kami pensiun EC2-Classic pada 15 Agustus 2022. Kami menyarankan Anda bermigrasi dari EC2-Classic ke VPC. [Untuk informasi selengkapnya, lihat Memigrasi dari EC2-Classic ke VPC di Panduan Pengguna Amazon EC2 atau Panduan Pengguna Amazon EC2](#). Lihat juga posting blog [EC2-Classic-Classic Networking is Retiring - Inilah Cara Mempersiapkan](#).

Buat grup keamanan, yang bertindak sebagai firewall virtual yang mengontrol lalu lintas jaringan untuk satu atau lebih instance EC2. Secara default, Amazon EC2 kaitkan instans Anda dengan grup keamanan yang tidak mengizinkan lalu lintas masuk. Anda dapat membuat grup keamanan yang memungkinkan instans EC2 Anda menerima lalu lintas tertentu. Misalnya, jika Anda perlu terhubung ke instance Linux, Anda harus mengkonfigurasi grup keamanan untuk mengizinkan lalu lintas SSH. Anda dapat membuat grup keamanan menggunakan Amazon EC2 konsol atau AWS SDK for Java.

Anda membuat grup keamanan untuk digunakan di EC2-Classic atau EC2-VPC. Untuk informasi selengkapnya tentang EC2-Classic dan EC2-VPC, lihat [Platform yang Didukung](#) di Amazon EC2 Panduan Pengguna untuk Instans Linux.

Untuk informasi selengkapnya tentang membuat grup keamanan menggunakan Amazon EC2 konsol, lihat [Grup Amazon EC2 Keamanan](#) di Panduan Amazon EC2 Pengguna untuk Instans Linux.

1. Membuat dan menginisialisasi sebuah [CreateSecurityGroupRequest](#) instance. Gunakan [withGroupName](#) metode untuk mengatur nama grup keamanan, dan metode [withDescription](#) untuk [mengatur deskripsi](#) grup keamanan, sebagai berikut:

```
CreateSecurityGroupRequest csgr = new CreateSecurityGroupRequest();
csgr.withGroupName("JavaSecurityGroup").withDescription("My security group");
```

Nama grup keamanan harus unik di AWS wilayah tempat Anda menginisialisasi Amazon EC2 klien Anda. Anda harus menggunakan karakter US-ASCII untuk nama dan deskripsi grup keamanan.

2. Lulus objek permintaan sebagai parameter ke [createSecurityGroup](#) metode. Metode mengembalikan [CreateSecurityGroupResult](#) objek, sebagai berikut:

```
CreateSecurityGroupResult createSecurityGroupResult =
    amazonEC2Client.createSecurityGroup(csgr);
```

Jika Anda mencoba membuat grup keamanan dengan nama yang sama dengan grup keamanan yang ada [createSecurityGroup](#), berikan pengecualian.

Secara default, grup keamanan baru tidak mengizinkan lalu lintas masuk ke Amazon EC2 instans Anda. Untuk mengizinkan lalu lintas masuk, Anda harus secara eksplisit mengotorisasi masuknya grup keamanan. Anda dapat mengotorisasi ingress untuk alamat IP individual, untuk berbagai alamat IP, untuk protokol tertentu, dan untuk port TCP/UDP.

1. Membuat dan menginisialisasi sebuah [IpPermission](#) instance. Gunakan metode [withIPv4Ranges](#) untuk mengatur rentang alamat IP untuk mengotorisasi masuknya, dan gunakan [withIpProtocol](#) metode untuk mengatur protokol IP. Gunakan [withToPort](#) metode [withFromPort](#) dan untuk menentukan rentang port untuk mengotorisasi masuknya, sebagai berikut:

```
IpPermission ipPermission =
    new IpPermission();

IpRange ipRange1 = new IpRange().withCidrIp("111.111.111.111/32");
IpRange ipRange2 = new IpRange().withCidrIp("150.150.150.150/32");

ipPermission.withIPv4Ranges(Arrays.asList(new IpRange[] {ipRange1, ipRange2}))
    .withIpProtocol("tcp")
    .withFromPort(22)
    .withToPort(22);
```

Semua kondisi yang Anda tentukan dalam `IpPermission` objek harus dipenuhi agar masuknya diizinkan.

Tentukan alamat IP menggunakan notasi CIDR. Jika Anda menentukan protokol sebagai TCP/UDP, Anda harus menyediakan port sumber dan port tujuan. Anda dapat mengotorisasi port hanya jika Anda menentukan TCP atau UDP.

2. Membuat dan menginisialisasi sebuah [AuthorizeSecurityGroupIngressRequest](#) instance. Gunakan `withGroupName` metode untuk menentukan nama grup keamanan, dan teruskan `IpPermission` objek yang Anda inisialisasi sebelumnya ke [withIpPermissions](#) metode, sebagai berikut:

```
AuthorizeSecurityGroupIngressRequest authorizeSecurityGroupIngressRequest =
    new AuthorizeSecurityGroupIngressRequest();

authorizeSecurityGroupIngressRequest.withGroupName("JavaSecurityGroup")
    .withIpPermissions(ipPermission);
```

3. Lulus objek permintaan ke dalam metode [authorizeSecurityGroupIngress](#), sebagai berikut:

```
amazonEC2Client.authorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);
```

Jika Anda memanggil `authorizeSecurityGroupIngress` dengan alamat IP yang masuknya sudah diotorisasi, metode ini melempar pengecualian. Buat dan inisialisasi `IpPermission` objek baru untuk mengotorisasi ingress untuk IP, port, dan protokol yang berbeda sebelum menelepon `AuthorizeSecurityGroupIngress`.

Setiap kali Anda memanggil metode [authorizeSecurityGroupIngress](#) atau [authorizeSecurityGroupEgress](#), aturan ditambahkan ke grup keamanan Anda.

Membuat Pasangan Kunci

Anda harus menentukan key pair saat meluncurkan instans EC2 dan kemudian menentukan kunci privat saat terhubung ke instans. Anda dapat membuat key pair atau menggunakan key pair yang sudah ada yang Anda gunakan saat meluncurkan instans lainnya. Untuk informasi selengkapnya, lihat [Amazon EC2 Pasangan kunci](#) di dalam [Amazon EC2 Panduan Pengguna untuk Instans Linux](#).

1. Membuat dan menginisialisasi [CreateKeyPairRequest](#) contoh. Menggunakan [withKeyName](#) metode untuk mengatur nama key pair, sebagai berikut:

```
CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();  
createKeyPairRequest.withKeyName(keyName);
```

Important

Nama pasangan harus unik. Jika Anda mencoba membuat key pair dengan nama kunci yang sama dengan key pair yang ada, Anda akan mendapatkan pengecualian.

2. Lulus objek permintaan ke [CreateKeyPair](#) metode. Metode mengembalikan [CreateKeyPairResult](#) misalnya, sebagai berikut:

```
CreateKeyPairResult createKeyPairResult =  
amazonEC2Client.createKeyPair(createKeyPairRequest);
```

3. Panggil objek hasil [getKeyPair](#) metode untuk mendapatkan [keyPair](#) objek. Memanggil [getKeyPair](#) objek [getKeyMaterial](#) metode untuk mendapatkan terenkripsi kunci pribadi PEM-dikodekan, sebagai berikut:

```
KeyPair keyPair = new KeyPair();  
  
keyPair = createKeyPairResult.getKeyPair();  
  
String privateKey = keyPair.getKeyMaterial();
```

Jalankan Amazon EC2 Instans

Gunakan prosedur berikut untuk meluncurkan satu atau lebih instans EC2 yang dikonfigurasi secara identik dari Amazon Machine Image (AMI) yang sama. Setelah membuat instans EC2, Anda dapat memeriksa statusnya. Setelah instans EC2 berjalan, Anda dapat terhubung ke instans EC2.

1. Membuat dan menginisialisasi [RunInstancesRequest](#) contoh. Pastikan AMI, key pair, dan grup keamanan yang Anda tetapkan ada di wilayah yang Anda tentukan saat membuat objek klien.

```
RunInstancesRequest runInstancesRequest =  
    new RunInstancesRequest();  
  
runInstancesRequest.withImageId("ami-a9d09ed1")
```

```
.withInstanceType(InstanceType.T1Micro)
.withMinCount(1)
.withMaxCount(1)
.withKeyName("my-key-pair")
.withSecurityGroups("my-security-group");
```

withImageId

- ID AMI. Untuk mempelajari cara menemukan AMI publik yang disediakan oleh Amazon atau membuat milik Anda sendiri, lihat [Amazon Machine Image \(AMI\)](#).

withInstanceType

- Jenis instans yang kompatibel dengan AMI yang ditentukan. Untuk informasi selengkapnya, lihat [Tipe instancediAmazon EC2Panduan Pengguna untuk Instans Linux](#).

withMinCount

- Jumlah instans EC2 untuk diluncurkan. Jika ini lebih banyak contoh daripadaAmazon EC2dapat meluncurkan di Target Availability Zone,Amazon EC2meluncurkan tidak ada contoh.

withMaxCount

- Jumlah maksimum instans EC2 untuk diluncurkan. Jika ini lebih banyak contoh daripadaAmazon EC2dapat meluncurkan di Target Availability Zone,Amazon EC2meluncurkan jumlah terbesar yang mungkin dari contoh di atasMinCount. Anda dapat meluncurkan antara 1 dan jumlah maksimum instans yang diizinkan untuk jenis instans. Untuk informasi selengkapnya, lihat [Berapa banyak instans yang dapat saya jalankanAmazon EC2diAmazon EC2FAQ Umum](#).

withKeyName

- Nama key pair EC2. Jika Anda meluncurkan instans tanpa menentukan key pair, Anda tidak dapat terhubung dengannya. Untuk informasi selengkapnya, lihat [Buat Pasangan Kunci](#).

withSecurityGroups

- Satu atau lebih grup keamanan. Untuk informasi selengkapnya, lihat [BuatAmazon EC2Grup Keamanan](#).

2. Luncurkan instance dengan melewati objek permintaan ke [runInstances](#) metode. Metode mengembalikan [RunInstancesResult](#) objek, sebagai berikut:

```
RunInstancesResult result = amazonEC2Client.runInstances(
    runInstancesRequest);
```

Setelah instans Anda berjalan, Anda dapat terhubung dengannya menggunakan key pair Anda. Untuk informasi selengkapnya, lihat [Connect ke Instans Linux Anda](#) di Amazon EC2 Panduan Pengguna untuk Instans Linux.

Menggunakan Peran IAM untuk Memberikan Akses ke AWS Sumber Daya di Amazon EC2

Semua permintaan untuk Amazon Web Services (AWS) harus ditandatangani secara kriptografis menggunakan kredensi yang dikeluarkan oleh AWS. Anda dapat menggunakan IAM role untuk dengan mudah memberikan akses yang aman ke AWS sumber daya dari Amazon EC2 contoh.

Topik ini memberikan informasi tentang cara menggunakan peran IAM dengan aplikasi Java SDK berjalan Amazon EC2. Untuk informasi selengkapnya tentang instans IAM, lihat [Peran IAM untuk Amazon EC2](#) di dalam Amazon EC2 Panduan Pengguna untuk Instans Linux.

Rantai penyedia default dan profil instans EC2

Jika aplikasi Anda membuat AWS klien menggunakan konstruktor default, maka klien akan mencari kredensia menggunakan rantai penyedia kredensia default, dalam urutan sebagai berikut:

1. Dalam properti sistem Java: `aws.accessKeyId` dan `aws.secretKey`.
2. Dalam variabel lingkungan sistem: `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY`.
3. Dalam file kredensia default (lokasi file ini bervariasi menurut platform).
4. Kredensia disampaikan melalui Amazon EC2 layanan kontainer jika `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` variabel lingkungan diatur dan manajer keamanan memiliki izin untuk mengakses variabel.
5. Dikredensia profil instans, yang ada dalam metadata instance yang terkait dengan peran IAM untuk instans EC2.
6. Kredensia Token Identity Web dari lingkungan atau wadah.

Parameter kredensia profil instans langka dalam rantai penyedia default hanya tersedia saat menjalankan aplikasi Anda pada Amazon EC2 misalnya, tetapi memberikan kemudahan terbesar penggunaan dan keamanan terbaik ketika bekerja dengan Amazon EC2 contoh. Anda juga dapat melewati sebuah [InstanceProfileCredentialsProvider](#) contoh langsung ke konstruktor klien untuk mendapatkan kredensia profil misalnya tanpa melanjutkan melalui seluruh rantai penyedia default.

Misalnya:


```
AmazonS3 s3 = AmazonS3ClientBuilder.standard()
    .withCredentials(new InstanceProfileCredentialsProvider(false))
    .build();
```

Saat menggunakan pendekatan ini, SDK mengambil sementara AWS kredensial yang memiliki izin yang sama dengan yang terkait dengan peran IAM yang terkait dengan Amazon EC2 misalnya dalam profil instans. Meskipun kredensial ini bersifat sementara dan akhirnya akan berakhir, `InstanceProfileCredentialsProvider` secara berkala menyegarkan mereka untuk Anda sehingga kredensial yang diperoleh terus memungkinkan akses ke AWS.

Important

Penyegaran kredensial otomatis terjadi ketika Anda menggunakan konstruktor klien default, yang menciptakan sendiri `InstanceProfileCredentialsProvider` sebagai bagian dari rantai penyedia default, atau ketika Anda melewati `InstanceProfileCredentialsProvider` contoh langsung ke konstruktor klien. Jika Anda menggunakan metode lain untuk mendapatkan atau meneruskan kredensial profil instans, Anda bertanggung jawab untuk memeriksa dan menyegarkan kredensial kedaluwarsa.

Jika konstruktor klien tidak dapat menemukan kredensial menggunakan rantai penyedia kredensial, itu akan melempar [AmazonClientException](#).

Panduan: Menggunakan peran IAM untuk instans EC2

Panduan berikut menunjukkan kepada Anda cara untuk mengambil objek dari Amazon S3 menggunakan peran IAM untuk mengelola akses.

Buat IAM Role

Buat peran IAM yang memberikan akses hanya baca Amazon S3.

1. Buka [konsol IAM](#).
2. Di panel navigasi, pilih **Peran** maka **Buat Peran Baru**.
3. Masukkan nama peran, lalu pilih **Langkah Selanjutnya**. Ingat nama ini, karena Anda akan membutuhkannya ketika Anda meluncurkan Amazon EC2 contoh.
4. Pada **Pilih Tipe Peran** halaman, di bawah **Layanan AWS Peran**, pilih **Amazon EC2**.

5. Pada **Mengatur Izin halaman**, di bawah **Pilih Templat Kebijakan**, pilih **Amazon S3 Akses Hanya** **Baca** **Langkah Selanjutnya**.
6. Pada **Tinjau halaman**, pilih **Buat Peran**.

Luncurkan Instans EC2 dan Tentukan Peran IAM Anda

Anda dapat meluncurkan Amazon EC2 instans dengan peran IAM menggunakan Amazon EC2 konsol atau AWS SDK for Java.

- Untuk meluncurkan sebuah Amazon EC2 instans menggunakan konsol, ikuti petunjuk dalam [Memulai dengan Amazon EC2 Instans Linux](#) di dalam Amazon EC2 Panduan Pengguna untuk Instans Linux.

Ketika Anda mencapai **Meninjau Peluncuran instans** halaman, pilih **Mengedit detail instans**. Masuk **Peran IAM**, pilih peran IAM yang Anda buat sebelumnya. Selesaikan prosedur sesuai petunjuk.

Note

Anda harus membuat atau menggunakan grup keamanan dan key pair yang ada untuk terhubung ke instance.

- Untuk meluncurkan sebuah Amazon EC2 instans dengan peran IAM menggunakan AWS SDK for Java, lihat [Jalankan sebuah Amazon EC2 Instans](#).

Buat Aplikasi Anda

Mari kita membangun aplikasi sampel untuk berjalan pada instans EC2. Pertama, buat direktori yang dapat Anda gunakan untuk menyimpan file tutorial Anda (misalnya, `GetS3ObjectApp`).

Selanjutnya, salin AWS SDK for Java perpustakaan ke direktori yang baru dibuat. Jika Anda mengunduh AWS SDK for Java ke `~/Downloads` direktori, Anda dapat menyalinnya menggunakan perintah berikut:

```
cp -r ~/Downloads/aws-java-sdk-{1.7.5}/lib .
cp -r ~/Downloads/aws-java-sdk-{1.7.5}/third-party .
```

Buka file baru, sebut saja `GetS3Object.java`, dan tambahkan kode berikut:

```
import java.io.*;

import com.amazonaws.auth.*;
import com.amazonaws.services.s3.*;
import com.amazonaws.services.s3.model.*;
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;

public class GetS3Object {
    private static final String bucketName = "text-content";
    private static final String key = "text-object.txt";

    public static void main(String[] args) throws IOException
    {
        AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

        try {
            System.out.println("Downloading an object");
            S3Object s3object = s3Client.getObject(
                new GetObjectRequest(bucketName, key));
            displayTextInputStream(s3object.getObjectContent());
        }
        catch(AmazonServiceException ase) {
            System.err.println("Exception was thrown by the service");
        }
        catch(AmazonClientException ace) {
            System.err.println("Exception was thrown by the client");
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException
    {
        // Read one text line at a time and display.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        while(true)
        {
            String line = reader.readLine();
            if(line == null) break;
            System.out.println( "    " + line );
        }
        System.out.println();
    }
}
```

Buka file baru, sebut `sajabuild.xml`, dan tambahkan baris berikut:

```
<project name="Get {S3} Object" default="run" basedir=".">
  <path id="aws.java.sdk.classpath">
    <fileset dir="./lib" includes="**/*.jar"/>
    <fileset dir="./third-party" includes="**/*.jar"/>
    <pathelement location="lib"/>
    <pathelement location="."/>
  </path>

  <target name="build">
    <javac debug="true"
      includeantruntime="false"
      srcdir="."
      destdir="."
      classpathref="aws.java.sdk.classpath"/>
  </target>

  <target name="run" depends="build">
    <java classname="GetS3Object" classpathref="aws.java.sdk.classpath" fork="true"/>
  </target>
</project>
```

Bangun dan jalankan program yang dimodifikasi. Perhatikan bahwa tidak ada kredensi yang disimpan dalam program. Oleh karena itu, kecuali Anda memiliki `AndaAWSKredensia` yang ditentukan sudah, kode akan melempar `AmazonServiceException`. Misalnya:

```
$ ant
Buildfile: /path/to/my/GetS3ObjectApp/build.xml

build:
[javac] Compiling 1 source file to /path/to/my/GetS3ObjectApp

run:
[java] Downloading an object
[java] AmazonServiceException

BUILD SUCCESSFUL
```

Transfer Program yang Disusun ke Instans EC2 Anda

Transfer program ke Amazon EC2 contoh menggunakan salinan aman (), bersama dengan AWS SDK for Java perpustakaan. Urutan perintah terlihat seperti berikut ini.

```
scp -p -i {my-key-pair}.pem GetS3Object.class ec2-user@{public_dns}:GetS3Object.class
scp -p -i {my-key-pair}.pem build.xml ec2-user@{public_dns}:build.xml
scp -r -p -i {my-key-pair}.pem lib ec2-user@{public_dns}:lib
scp -r -p -i {my-key-pair}.pem third-party ec2-user@{public_dns}:third-party
```

Note

Tergantung pada distribusi Linux yang Anda gunakan, Nama Pengguna mungkin “ec2-user”, “root”, atau “ubuntu”. Untuk mendapatkan nama DNS publik dari instans Anda, buka [Konsol EC2](#) dan carilah DNS Publik value dalam Deskripsi tab (misalnya, ec2-198-51-100-1.compute-1.amazonaws.com).

Dalam perintah sebelumnya:

- `GetS3Object.class` adalah program Anda dikompilasi
- `build.xml` adalah file semut yang digunakan untuk membangun dan menjalankan program Anda
- `lib` dan `third-party` direktori adalah folder pustaka yang sesuai dari AWS SDK for Java.
- Parameter `-r` switch menunjukkan bahwa `scp` harus melakukan salinan rekursif dari semua isi `library` dan `third-party` direktori dalam AWS SDK for Java distribusi.
- Parameter `-p` switch menunjukkan bahwa `scp` harus menjaga izin dari file sumber ketika salinan mereka ke tujuan.

Note

Parameter `-p` switch hanya berfungsi di Linux, macOS, atau Unix. Jika Anda menyalin file dari Windows, Anda mungkin perlu memperbaiki izin file pada instans Anda menggunakan perintah berikut:

```
chmod -R u+rwX GetS3Object.class build.xml lib third-party
```

Jalankan Program Sampel pada Instans EC2

Untuk menjalankan program, hubungkan ke [Amazon EC2 contoh](#). Untuk informasi selengkapnya, lihat [Connect ke Instans Linux Anda](#) di dalam [Amazon EC2 Panduan Pengguna untuk Instans Linux](#).

Jika `ant` tidak tersedia pada instans Anda, instal menggunakan perintah berikut:

```
sudo yum install ant
```

Kemudian, jalankan program menggunakan `ant` sebagai berikut:

```
ant run
```

Program ini akan menulis isi dari `Amazon S3` ke `beratan` jendela perintah Anda.

Tutorial: Amazon EC2 Instans Spot

Gambaran Umum

Instans Spot memungkinkan Anda untuk menawar pada tidak digunakan Amazon Elastic Compute Cloud (Amazon EC2) kapasitas hingga 90% dibandingkan harga Instans Pesanan dan menjalankan instans yang diperoleh selama tawaran Anda melebihi saat ini Harga Spot. Amazon EC2 mengubah Harga Spot secara berkala berdasarkan penawaran dan permintaan, dan pelanggan yang menawarkannya memenuhi atau melampaui itu mendapatkan akses ke Instans Spot yang tersedia. Seperti Instans Sesuai Permintaan dan Instans Cadangan, Instans Spot memberi Anda opsi lain untuk mendapatkan lebih banyak kapasitas komputasi.

Instans Spot dapat secara signifikan menurunkan Amazon EC2 biaya untuk pemrosesan batch, penelitian ilmiah, pemrosesan gambar, pengkodean video, data dan perayapan web, analisis keuangan, dan pengujian. Selain itu, Instans Spot memberi Anda akses ke kapasitas tambahan dalam jumlah besar dalam situasi di mana kebutuhan akan kapasitas tersebut tidak mendesak.

Untuk menggunakan Instans Spot, tempatkan permintaan Instans Spot yang menentukan harga maksimum yang bersedia Anda bayarkan per jam instans; ini tawaran Anda. Jika tawaran Anda melebihi Harga Spot saat ini, permintaan Anda terpenuhi dan instans Anda akan berjalan sampai Anda memilih untuk menghentikannya atau Harga Spot meningkat di atas tawaran Anda (mana yang lebih cepat).

Penting untuk dicatat:

- Anda akan sering membayar kurang per jam dari tawaran Anda. Amazon EC2 menyesuaikan Harga Spot secara berkala saat permintaan masuk dan perubahan pasokan yang tersedia. Setiap orang membayar Harga Spot yang sama untuk periode tersebut terlepas dari apakah tawaran mereka lebih tinggi. Oleh karena itu, Anda mungkin membayar kurang dari tawaran Anda, tetapi Anda tidak akan pernah membayar lebih dari tawaran Anda.
- Jika Anda menjalankan Instans Spot dan tawaran Anda tidak lagi memenuhi atau melebihi Harga Spot saat ini, instans Anda akan dihentikan. Ini berarti bahwa Anda akan ingin memastikan bahwa beban kerja dan aplikasi Anda cukup fleksibel untuk memanfaatkan kapasitas oportunistik ini.

Instans Spot berkinerja persis seperti lainnya Amazon EC2 instance saat berjalan, dan seperti lainnya Amazon EC2 Instans Spot dapat dihentikan ketika Anda tidak lagi membutuhkannya. Jika Anda mengakhiri instans, Anda membayar sebagian jam yang digunakan (seperti yang Anda lakukan untuk Instans Pesanan atau Cadangan). Namun, jika Harga Spot berjalan di atas bid Anda dan instans Anda dihentikan oleh Amazon EC2, Anda tidak akan dikenakan biaya untuk sebagian jam penggunaan.

Tutorial ini menunjukkan cara menggunakan AWS SDK for Java untuk melakukan hal berikut.

- Kirim Permintaan Spot
- Tentukan kapan Spot Request terpenuhi
- Batalkan Permintaan Spot
- Akhiri instans terkait

Prasyarat

Untuk menggunakan tutorial ini Anda harus memiliki AWS SDK for Java diinstal, serta telah memenuhi prasyarat instalasi dasar. Lihat [Siapkan perangkat AWS SDK for Java](#) untuk informasi lebih lanjut.

Langkah 1: Menyiapkan Kredensi Anda

Untuk mulai menggunakan contoh kode ini, Anda perlu mengatur AWS kredensi.

Lihat [Mengatur AWS Kredensial dan Wilayah untuk Pembangunan](#) untuk petunjuk tentang cara melakukannya.

Note

Kami menyarankan Anda menggunakan kredensi pengguna IAM untuk memberikan nilai-nilai ini. Untuk informasi selengkapnya, lihat [Mendaftar keAWSBuat Pengguna IAM](#).

Setelah Anda mengonfigurasi pengaturan Anda, Anda dapat mulai menggunakan kode dalam contoh.

Langkah 2: Menyiapkan Grup Keamanan

SEBUAH grup keamanan bertindak sebagai firewall yang mengontrol lalu lintas yang diizinkan masuk dan keluar dari sekelompok instans. Secara default, sebuah instance dimulai tanpa grup keamanan apa pun, yang berarti bahwa semua lalu lintas IP yang masuk, pada port TCP mana pun akan ditolak. Jadi, sebelum mengirimkan Permintaan Spot kami, kami akan menyiapkan grup keamanan yang memungkinkan lalu lintas jaringan yang diperlukan. Untuk tujuan tutorial ini, kita akan membuat grup keamanan baru yang disebut "GettingStarted" yang memungkinkan lalu lintas Secure Shell (SSH) dari alamat IP tempat Anda menjalankan aplikasi Anda. Untuk menyiapkan grup keamanan baru, Anda harus menyertakan atau menjalankan contoh kode berikut yang mengatur grup keamanan secara terprogram.

Setelah kita membuat AmazonEC2 objek klien, kita membuat CreateSecurityGroupRequest objek dengan nama, "GettingStarted" dan deskripsi untuk grup keamanan. Lalu kita sebut `ec2.createSecurityGroupAPI` untuk membuat grup.

Untuk mengaktifkan akses ke grup, kita membuat `ipPermission` objek dengan rentang alamat IP diatur ke representasi CIDR dari subnet untuk komputer lokal; akhiran "/10" pada alamat IP menunjukkan subnet untuk alamat IP yang ditentukan. Kami juga mengkonfigurasi `ipPermission` objek dengan protokol TCP dan port 22 (SSH). Langkah terakhir adalah memanggil `ec2.authorizeSecurityGroupIngress` dengan nama grup keamanan kami dan `ipPermission` objek.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Create a new security group.
try {
    CreateSecurityGroupRequest securityGroupRequest = new
    CreateSecurityGroupRequest("GettingStartedGroup", "Getting Started Security Group");
    ec2.createSecurityGroup(securityGroupRequest);
}
```



```
} catch (AmazonServiceException ase) {
    // Likely this means that the group is already created, so ignore.
    System.out.println(ase.getMessage());
}

String ipAddr = "0.0.0.0/0";

// Get the IP of the current host, so that we can limit the Security
// Group by default to the ip range associated with your subnet.
try {
    InetAddress addr = InetAddress.getLocalHost();

    // Get IP Address
    ipAddr = addr.getHostAddress()+"/10";
} catch (UnknownHostException e) {
}

// Create a range that you would like to populate.
ArrayList<String> ipRanges = new ArrayList<String>();
ipRanges.add(ipAddr);

// Open up port 22 for TCP traffic to the associated IP
// from above (e.g. ssh traffic).
ArrayList<IpPermission> ipPermissions = new ArrayList<IpPermission> ();
IpPermission ipPermission = new IpPermission();
ipPermission.setIpProtocol("tcp");
ipPermission.setFromPort(new Integer(22));
ipPermission.setToPort(new Integer(22));
ipPermission.setIpRanges(ipRanges);
ipPermissions.add(ipPermission);

try {
    // Authorize the ports to the used.
    AuthorizeSecurityGroupIngressRequest ingressRequest =
        new AuthorizeSecurityGroupIngressRequest("GettingStartedGroup", ipPermissions);
    ec2.authorizeSecurityGroupIngress(ingressRequest);
} catch (AmazonServiceException ase) {
    // Ignore because this likely means the zone has
    // already been authorized.
    System.out.println(ase.getMessage());
}
```

Perhatikan bahwa Anda hanya perlu menjalankan aplikasi ini sekali untuk membuat grup keamanan baru.

Anda juga dapat membuat grup keamanan menggunakan perangkat AWS Toolkit for Eclipse. Lihat [Mengelola Grup Keamanan dari AWS Cost Explorer](#) untuk informasi lebih lanjut.

Langkah 3: Mengirimkan Permintaan Spot Anda

Untuk mengirimkan permintaan Spot, pertama-tama Anda harus menentukan jenis instans, Amazon Machine Image (AMI), dan harga tawaran maksimum yang ingin Anda gunakan. Anda juga harus menyertakan grup keamanan yang kami konfigurasi sebelumnya, sehingga Anda dapat masuk ke instance jika diinginkan.

Ada beberapa jenis instans untuk dipilih; kunjungi [Amazon EC2 Jenis instans untuk daftar lengkap](#). Untuk tutorial ini, kita akan menggunakan t1.micro, jenis instance termurah yang tersedia. Selanjutnya, kita akan menentukan jenis AMI yang ingin kita gunakan. Kami akan menggunakan ami-a9d09ed1, AMI Amazon Linux terbaru yang tersedia saat menulis tutorial ini. AMI terbaru dapat berubah dari waktu ke waktu, tetapi Anda selalu dapat menentukan versi terbaru AMI dengan mengikuti langkah-langkah berikut:

1. Buka [konsol Amazon EC2](#).
2. Pilih Luncurkan Instans tombol.
3. Jendela pertama menampilkan AMI yang tersedia. ID AMI tercantum di sebelah setiap judul AMI. Atau, Anda dapat menggunakan perangkat DescribeImages API, tetapi memanfaatkan perintah itu berada di luar cakupan tutorial ini.

Ada banyak cara untuk mendekati bidding untuk Instans Spot; untuk mendapatkan gambaran luas tentang berbagai pendekatan yang harus Anda lihat [Penawaran untuk Instans Spot](#) video. Namun, untuk memulai, kami akan menjelaskan tiga strategi umum: tawaran untuk memastikan biaya kurang dari harga sesuai permintaan; tawaran berdasarkan nilai perhitungan yang dihasilkan; tawaran untuk memperoleh kapasitas komputasi secepat mungkin.

- Kurangi Biaya di bawah Permintaan Anda memiliki pekerjaan pemrosesan batch yang akan memakan waktu beberapa jam atau hari untuk dijalankan. Namun, Anda fleksibel sehubungan dengan kapan dimulai dan ketika selesai. Anda ingin melihat apakah Anda dapat menyelesaikannya dengan biaya lebih rendah daripada Instans Pesanan. Anda memeriksa riwayat Harga Spot untuk jenis instans menggunakan salah satu [AWS Management Console](#) atau [Amazon EC2 API](#). Untuk informasi selengkapnya, kunjungi [Melihat Riwayat Harga Spot](#). Setelah

menganalisis riwayat harga untuk jenis instans yang Anda inginkan di Availability Zone tertentu, Anda memiliki dua pendekatan alternatif untuk tawaran Anda:

- Anda dapat menawar di ujung atas kisaran Harga Spot (yang masih di bawah harga Sesuai Permintaan), mengantisipasi bahwa permintaan Spot satu kali Anda kemungkinan besar akan terpenuhi dan dijalankan untuk waktu komputasi yang cukup berturut-turut untuk menyelesaikan pekerjaan.
- Atau, Anda dapat menentukan jumlah yang bersedia Anda bayar untuk Instans Spot sebagai % dari harga Instans Pesanan, dan berencana untuk menggabungkan banyak instans yang diluncurkan dari waktu ke waktu melalui permintaan persisten. Jika harga yang ditentukan terlampaui, maka Instans Spot akan berakhir. (Kami akan menjelaskan cara mengotomatisasi tugas ini nanti dalam tutorial ini.)
- Bayar Tidak Lebih dari Nilai Hasil Anda memiliki pekerjaan pemrosesan data untuk dijalankan. Anda memahami nilai hasil pekerjaan cukup baik untuk mengetahui berapa banyak mereka layak dalam hal biaya komputasi. Setelah Anda menganalisis riwayat Harga Spot untuk jenis instans Anda, Anda memilih harga penawaran di mana biaya waktu komputasi tidak lebih dari nilai hasil pekerjaan. Anda membuat tawaran persisten dan memungkinkannya berjalan sebentar-sebentar saat Harga Spot berfluktuasi pada atau di bawah tawaran Anda.
- Memperoleh Kapasitas Komputasi Cepat Anda memiliki kebutuhan jangka pendek yang tidak diantisipasi untuk kapasitas tambahan yang tidak tersedia melalui Instans Pesanan. Setelah Anda menganalisis riwayat Harga Spot untuk jenis instans Anda, Anda menawar di atas harga historis tertinggi untuk memberikan kemungkinan besar permintaan Anda akan terpenuhi dengan cepat dan melanjutkan komputasi hingga selesai.

Setelah memilih harga tawaran, Anda siap meminta Instans Spot. Untuk tujuan tutorial ini, kami akan menawar harga On-Demand (\$0.03) untuk memaksimalkan kemungkinan bahwa tawaran akan terpenuhi. Anda dapat menentukan jenis instans yang tersedia dan harga On-Demand untuk instans dengan membuka Amazon EC2 Halaman harga. Saat Instance Spot berjalan, Anda membayar harga Spot yang berlaku untuk jangka waktu instans Anda berjalan. Harga Instans Spot ditetapkan oleh Amazon EC2 dan menyesuaikan secara bertahap berdasarkan tren jangka panjang dalam penawaran dan permintaan untuk kapasitas Instans Spot. Anda juga dapat menentukan jumlah yang bersedia Anda bayar untuk Instans Spot sebagai % dari harga Instans Pesanan. Untuk meminta Instans Spot, Anda hanya perlu membuat permintaan Anda dengan parameter yang Anda pilih sebelumnya. Kita mulai dengan membuat `RequestSpotInstanceRequest` objek. Objek permintaan memerlukan jumlah instans yang ingin Anda mulai dan harga penawaran. Selain itu, Anda perlu mengatur perangkat `LaunchSpecification` untuk permintaan, yang mencakup jenis

instans, ID AMI, dan grup keamanan yang ingin Anda gunakan. Setelah permintaan diisi, Anda menghubungkan `requestSpotInstances` metode pada metode `AmazonEC2Client` objek. Contoh berikut menunjukkan cara meminta Instans Spot.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Setup the specifications of the launch. This includes the
// instance type (e.g. t1.micro) and the latest Amazon Linux
// AMI id available. Note, you should always use the latest
// Amazon Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specifications to the request.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(requestRequest);
```

Menjalankan kode ini akan meluncurkan Permintaan Instans Spot baru. Ada opsi lain yang dapat Anda gunakan untuk mengonfigurasi Permintaan Spot Anda. Untuk mempelajari selengkapnya, kunjungi [Tutorial: Lanjutan Amazon EC2 Manajemen permintaan Spot](#) atau [RequestSpotInstances](#) kelas di AWS SDK for Java Referensi API.

Note

Anda akan dikenakan biaya untuk Instans Spot apa pun yang benar-benar diluncurkan, jadi pastikan Anda membatalkan permintaan apa pun dan mengakhiri instans yang Anda luncurkan untuk mengurangi biaya terkait.

Langkah 4: Menentukan Status Permintaan Spot Anda

Selanjutnya, kita ingin membuat kode untuk menunggu sampai permintaan Spot mencapai status “aktif” sebelum melanjutkan ke langkah terakhir. Untuk menentukan status permintaan Spot kami, kami memilih [describeSpotInstanceRequests](#) metode untuk status ID permintaan Spot yang ingin kita pantau.

ID permintaan yang dibuat di Langkah 2 disematkan dalam menanggapi `requestSpotInstances` permintaan. Contoh kode berikut menunjukkan cara mengumpulkan ID permintaan dari `requestSpotInstances` respon dan menggunakannya untuk mengisi `ArrayList`.

```
// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(requestRequest);
List<SpotInstanceRequest> requestResponses = requestResult.getSpotInstanceRequests();

// Setup an arraylist to collect all of the request ids we want to
// watch hit the running state.
ArrayList<String> spotInstanceRequestIds = new ArrayList<String>();

// Add all of the request ids to the hashset, so we can determine when they hit the
// active state.
for (SpotInstanceRequest requestResponse : requestResponses) {
    System.out.println("Created Spot Request:
"+requestResponse.getSpotInstanceRequestId());
    spotInstanceRequestIds.add(requestResponse.getSpotInstanceRequestId());
}
```

Untuk memantau ID permintaan Anda, hubungi `describeSpotInstanceRequests` metode untuk menentukan keadaan permintaan. Kemudian loop sampai permintaan tidak dalam keadaan “terbuka”. Perhatikan bahwa kita memantau keadaan tidak “terbuka”, bukan keadaan, katakanlah, “aktif”, karena permintaan bisa langsung menuju “tertutup” jika ada masalah dengan argumen permintaan Anda. Contoh kode berikut memberikan rincian tentang bagaimana untuk menyelesaikan tugas ini.

```
// Create a variable that will track whether there are any
// requests still in the open state.
boolean anyOpen;

do {
    // Create the describeRequest object with all of the request ids
    // to monitor (e.g. that we started).
    DescribeSpotInstanceRequestsRequest describeRequest = new
DescribeSpotInstanceRequestsRequest();
    describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

    // Initialize the anyOpen variable to false - which assumes there
    // are no requests open unless we find one that is still open.
    anyOpen=false;

    try {
        // Retrieve all of the requests we want to monitor.
        DescribeSpotInstanceRequestsResult describeResult =
ec2.describeSpotInstanceRequests(describeRequest);
        List<SpotInstanceRequest> describeResponses =
describeResult.getSpotInstanceRequests();

        // Look through each request and determine if they are all in
        // the active state.
        for (SpotInstanceRequest describeResponse : describeResponses) {
            // If the state is open, it hasn't changed since we attempted
            // to request it. There is the potential for it to transition
            // almost immediately to closed or cancelled so we compare
            // against open instead of active.
            if (describeResponse.getState().equals("open")) {
                anyOpen = true;
                break;
            }
        }
    } catch (AmazonServiceException e) {
        // If we have an exception, ensure we don't break out of
        // the loop. This prevents the scenario where there was
        // blip on the wire.
        anyOpen = true;
    }

    try {
        // Sleep for 60 seconds.
```

```
        Thread.sleep(60*1000);
    } catch (Exception e) {
        // Do nothing because it woke up early.
    }
} while (anyOpen);
```

Setelah menjalankan kode ini, Permintaan Instans Spot Anda akan selesai atau akan gagal dengan kesalahan yang akan output ke layar. Dalam kedua kasus tersebut, kami dapat melanjutkan ke langkah berikutnya untuk membersihkan setiap permintaan yang aktif dan menghentikan setiap instans yang sedang berjalan.

Langkah 5: Membersihkan Permintaan dan Instans Spot

Terakhir, kita perlu membersihkan permintaan dan instans kita. Penting bagi Anda untuk membatalkan setiap permintaan yang belum tertunggak dan mengakhiri setiap contoh. Hanya membatalkan permintaan Anda tidak akan menghentikan instans Anda, yang berarti bahwa Anda akan terus membayarnya. Jika Anda menghentikan instans Anda, permintaan Spot Anda dapat dibatalkan, namun ada beberapa skenario seperti jika Anda menggunakan tawaran persisten saat menghentikan instans Anda tidaklah cukup untuk menghentikan permintaan Anda agar tidak terpenuhi kembali. Oleh karena itu, membatalkan setiap tawaran aktif dan menghentikan setiap instans yang sedang berjalan adalah praktik terbaik.

Kode berikut menunjukkan cara membatalkan permintaan Anda.

```
try {
    // Cancel requests.
    CancelSpotInstanceRequestsRequest cancelRequest =
        new CancelSpotInstanceRequestsRequest(spotInstanceRequestIds);
    ec2.cancelSpotInstanceRequests(cancelRequest);
} catch (AmazonServiceException e) {
    // Write out any exceptions that may have occurred.
    System.out.println("Error cancelling instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Response Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

Untuk mengakhiri instans yang luar biasa, Anda memerlukan ID instance yang terkait dengan permintaan yang memulainya. Contoh kode berikut mengambil kode asli kami untuk memantau

instance dan menambahkan `ArrayList` di mana kita menyimpan ID instance yang terkait dengan `describeInstance` tanggapan.

```
// Create a variable that will track whether there are any requests
// still in the open state.
boolean anyOpen;
// Initialize variables.
ArrayList<String> instanceIds = new ArrayList<String>();

do {
    // Create the describeRequest with all of the request ids to
    // monitor (e.g. that we started).
    DescribeSpotInstanceRequestsRequest describeRequest = new
DescribeSpotInstanceRequestsRequest();
    describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

    // Initialize the anyOpen variable to false, which assumes there
    // are no requests open unless we find one that is still open.
    anyOpen = false;

    try {
        // Retrieve all of the requests we want to monitor.
        DescribeSpotInstanceRequestsResult describeResult =
            ec2.describeSpotInstanceRequests(describeRequest);

        List<SpotInstanceRequest> describeResponses =
            describeResult.getSpotInstanceRequests();

        // Look through each request and determine if they are all
        // in the active state.
        for (SpotInstanceRequest describeResponse : describeResponses) {
            // If the state is open, it hasn't changed since we
            // attempted to request it. There is the potential for
            // it to transition almost immediately to closed or
            // cancelled so we compare against open instead of active.
            if (describeResponse.getState().equals("open")) {
                anyOpen = true; break;
            }
            // Add the instance id to the list we will
            // eventually terminate.
            instanceIds.add(describeResponse.getInstanceId());
        }
    } catch (AmazonServiceException e) {
```



```

    // If we have an exception, ensure we don't break out
    // of the loop. This prevents the scenario where there
    // was blip on the wire.
    anyOpen = true;
}

try {
    // Sleep for 60 seconds.
    Thread.sleep(60*1000);
} catch (Exception e) {
    // Do nothing because it woke up early.
}
} while (anyOpen);

```

Menggunakan ID instance, disimpan dalam `ArrayList`, mengakhiri instance yang berjalan menggunakan cuplikan kode berikut.

```

try {
    // Terminate instances.
    TerminateInstancesRequest terminateRequest = new
    TerminateInstancesRequest(instanceIds);
    ec2.terminateInstances(terminateRequest);
} catch (AmazonServiceException e) {
    // Write out any exceptions that may have occurred.
    System.out.println("Error terminating instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}

```

Membawa itu Semua Bersama

Untuk menyatukan semua ini, kami menyediakan pendekatan yang lebih berorientasi objek yang menggabungkan langkah-langkah sebelumnya yang kami tunjukkan: menginisialisasi Klien EC2, mengirimkan Permintaan Spot, menentukan kapan Permintaan Spot tidak lagi berada dalam keadaan terbuka, dan membersihkan permintaan Spot yang masih ada dan instans terkait. Kami membuat kelas yang disebut `Request` yang melakukan tindakan ini.

Kami juga membuat `GettingStartedApp` kelas, yang memiliki metode utama di mana kita melakukan panggilan fungsi tingkat tinggi. Secara khusus, kami menginisialisasi `Request` subjek

dijelaskan sebelumnya. Kami mengirimkan permintaan Instans Spot. Kemudian kita menunggu permintaan Spot mencapai status “Aktif”. Akhirnya, kami membersihkan permintaan dan contoh.

Kode sumber lengkap untuk contoh ini dapat dilihat atau diunduh di [GitHub](#).

Selamat! Anda baru saja menyelesaikan tutorial memulai untuk mengembangkan perangkat lunak Spot Instance dengan AWS SDK for Java.

Langkah Selanjutnya

Lanjutkan dengan [Tutorial: Lanjutan Amazon EC2 Manajemen permintaan Spot](#).

Tutorial: Lanjutan Amazon EC2 Manajemen permintaan spot

Amazon EC2 Instans Spot memungkinkan Anda untuk menawar pada yang tidak digunakan Amazon EC2 kapasitas dan menjalankan instans tersebut selama tawaran Anda melebihi saat ini harga spot. Amazon EC2 mengubah harga spot secara berkala berdasarkan penawaran dan permintaan. Untuk informasi selengkapnya tentang Instans Spot Instans, lihat [Spot Instance](#) di dalam Amazon EC2 Panduan Pengguna untuk Instans Linux.

Prasyarat

Untuk menggunakan tutorial ini Anda harus memiliki AWS SDK for Java diinstal, serta telah memenuhi prasyarat instalasi dasar. Lihat [Menyiapkan AWS SDK for Java](#) Untuk informasi lebih lanjut.

Menyiapkan kredenensi Anda

Untuk mulai menggunakan sampel kode ini, Anda perlu mengatur AWS kredensi. Lihat [Mengatur AWS Kredensi dan Wilayah untuk Pembangunan](#) untuk petunjuk tentang cara melakukannya.

Note

Sebaiknya Anda menggunakan kredensi IAM pengguna untuk memberikan nilai-nilai ini. Untuk informasi selengkapnya, lihat [Mendaftar AWS dan Buat IAM Pengguna](#).

Setelah Anda mengkonfigurasi pengaturan Anda, Anda dapat memulai menggunakan kode dalam contoh.

Menyiapkan grup keamanan

Grup keamanan bertindak sebagai firewall yang mengontrol lalu lintas yang diizinkan masuk dan keluar dari grup instans. Secara default, sebuah instance dimulai tanpa grup keamanan apa pun, yang berarti bahwa semua lalu lintas IP yang masuk, pada port TCP mana pun akan ditolak. Jadi, sebelum mengirimkan Permintaan Spot kami, kami akan menyiapkan grup keamanan yang memungkinkan lalu lintas jaringan yang diperlukan. Untuk tujuan tutorial ini, kita akan membuat grup keamanan baru yang disebut "GettingStarted" yang memungkinkan lalu lintas Secure Shell (SSH) dari alamat IP tempat Anda menjalankan aplikasi Anda. Untuk menyiapkan grup keamanan baru, Anda harus menyertakan atau menjalankan contoh kode berikut yang mengatur grup keamanan secara terprogram.

Setelah kita membuat `AmazonEC2` objek klien, kita membuat `CreateSecurityGroupRequest` objek dengan nama, "gettingStarted" dan deskripsi untuk grup keamanan. Lalu kita sebut `ec2.createSecurityGroupAPI` untuk membuat grup.

Untuk mengaktifkan akses ke grup, kita membuat `ipPermission` objek dengan rentang alamat IP diatur ke representasi CIDR dari subnet untuk komputer lokal; akhiran "/10" pada alamat IP menunjukkan subnet untuk alamat IP yang ditentukan. Kami juga mengkonfigurasi `ipPermission` objek dengan protokol TCP dan port 22 (SSH). Langkah terakhir adalah memanggil `ec2.authorizeSecurityGroupIngress` dengan nama grup keamanan kami dan `ipPermission` objek.

(Kode berikut adalah sama dengan apa yang kita gunakan dalam tutorial pertama.)

```
// Create the AmazonEC2Client object so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.standard()
    .withCredentials(credentials)
    .build();

// Create a new security group.
try {
    CreateSecurityGroupRequest securityGroupRequest =
        new CreateSecurityGroupRequest("GettingStartedGroup",
            "Getting Started Security Group");
    ec2.createSecurityGroup(securityGroupRequest);
} catch (AmazonServiceException ase) {
    // Likely this means that the group is already created, so ignore.
    System.out.println(ase.getMessage());
}
```

```
String ipAddr = "0.0.0.0/0";

// Get the IP of the current host, so that we can limit the Security Group
// by default to the ip range associated with your subnet.
try {
    // Get IP Address
    InetAddress addr = InetAddress.getLocalHost();
    ipAddr = addr.getHostAddress()+"/10";
}
catch (UnknownHostException e) {
    // Fail here...
}

// Create a range that you would like to populate.
ArrayList<String> ipRanges = new ArrayList<String>();
ipRanges.add(ipAddr);

// Open up port 22 for TCP traffic to the associated IP from
// above (e.g. ssh traffic).
ArrayList<IpPermission> ipPermissions = new ArrayList<IpPermission> ();
IpPermission ipPermission = new IpPermission();
ipPermission.setIpProtocol("tcp");
ipPermission.setFromPort(new Integer(22));
ipPermission.setToPort(new Integer(22));
ipPermission.setIpRanges(ipRanges);
ipPermissions.add(ipPermission);

try {
    // Authorize the ports to the used.
    AuthorizeSecurityGroupIngressRequest ingressRequest =
        new AuthorizeSecurityGroupIngressRequest(
            "GettingStartedGroup",ipPermissions);
    ec2.authorizeSecurityGroupIngress(ingressRequest);
}
catch (AmazonServiceException ase) {
    // Ignore because this likely means the zone has already
    // been authorized.
    System.out.println(ase.getMessage());
}
```

Anda dapat melihat seluruh contoh kode ini

di `advanced.CreateSecurityGroupApp.javasampel` kode. Perhatikan bahwa Anda hanya perlu menjalankan aplikasi ini sekali untuk membuat grup keamanan baru.

Note

Anda juga dapat membuat grup keamanan menggunakan AWS Toolkit for Eclipse. Lihat [Mengelola Grup Keamanan dari AWS Cost Explorer](#) di dalam AWS Toolkit for Eclipse Panduan Pengguna untuk informasi lebih lanjut.

Opsi pembuatan permintaan Instans Spot Instance

Seperti yang kita jelaskan di [Tutorial: Amazon EC2 Spot Instance](#), Anda perlu membuat permintaan Anda dengan tipe instans, Amazon Machine Image (AMI), dan harga bid maksimum.

Mari kita mulai dengan membuat `RequestSpotInstanceRequest` objek. Objek permintaan memerlukan jumlah instans yang Anda inginkan dan harga penawaran. Selain itu, kita perlu mengatur `LaunchSpecification` untuk permintaan, yang mencakup jenis instans, ID AMI, dan grup keamanan yang ingin Anda gunakan. Setelah permintaan diisi, kami memanggil `requestSpotInstances` metode pada metode `AmazonEC2Client` objek. Contoh cara meminta Instans Spot berikut.

(Kode berikut adalah sama dengan apa yang kita gunakan dalam tutorial pertama.)

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set up the specifications of the launch. This includes the
// instance type (e.g. t1.micro) and the latest Amazon Linux
// AMI id available. Note, you should always use the latest
// Amazon Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
```

```
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Permintaan persisten vs satu kali

Saat membuat permintaan Spot, Anda dapat menentukan beberapa parameter opsional. Yang pertama adalah apakah permintaan Anda hanya satu kali atau persisten. Secara default, ini adalah permintaan satu kali. Permintaan satu kali hanya dapat dipenuhi sekali, dan setelah instans yang diminta dihentikan, permintaan akan ditutup. Permintaan persisten dipertimbangkan untuk pemenuhan setiap kali tidak ada Instans Spot yang berjalan untuk permintaan yang sama. Untuk menentukan jenis permintaan, Anda hanya perlu mengatur Type pada permintaan Spot. Hal ini dapat dilakukan dengan kode berikut.

```
// Retrieves the credentials from an AWSCredentials.properties file.
AWSCredentials credentials = null;
try {
    credentials = new PropertiesCredentials(
        GettingStartedApp.class.getResourceAsStream("AwsCredentials.properties"));
}
catch (IOException e1) {
    System.out.println(
        "Credentials were not properly entered into AwsCredentials.properties.");
    System.out.println(e1.getMessage());
    System.exit(-1);
}

// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest =
    new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
```

```
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set the type of the bid to persistent.
requestRequest.setType("persistent");

// Set up the specifications of the launch. This includes the
// instance type (e.g. t1.micro) and the latest Amazon Linux
// AMI id available. Note, you should always use the latest
// Amazon Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Membatasi durasi permintaan

Anda juga dapat menentukan jangka waktu permintaan Anda akan tetap valid. Anda dapat menentukan waktu mulai dan akhir untuk periode ini. Secara default, permintaan Spot akan dipertimbangkan untuk pemenuhan dari saat dibuat sampai permintaan tersebut terpenuhi atau dibatalkan oleh Anda. Namun Anda dapat membatasi masa berlaku jika Anda perlu. Contoh cara menentukan periode ini ditampilkan dalam kode berikut.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));
```

```
// Set the valid start time to be two minutes from now.
Calendar cal = Calendar.getInstance();
cal.add(Calendar.MINUTE, 2);
requestRequest.setValidFrom(cal.getTime());

// Set the valid end time to be two minutes and two hours from now.
cal.add(Calendar.HOUR, 2);
requestRequest.setValidUntil(cal.getTime());

// Set up the specifications of the launch. This includes
// the instance type (e.g. t1.micro)

// and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon
// Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType("t1.micro");

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(requestRequest);
```

Mengelompokkan Amazon EC2 Permintaan Instans Spot

Anda memiliki opsi untuk mengelompokkan permintaan Instans Spot Anda dengan beberapa cara yang berbeda. Kita akan melihat manfaat menggunakan grup peluncuran, grup Availability Zone, dan grup penempatan.

Jika Anda ingin memastikan Instans Spot Anda semua diluncurkan dan dihentikan bersama, maka Anda memiliki opsi untuk memanfaatkan grup peluncuran. Sebuah grup peluncuran adalah label yang mengelompokkan satu set tawaran bersama-sama. Semua instance dalam grup peluncuran dimulai dan diakhiri bersama. Catatan, jika instance dalam grup peluncuran telah terpenuhi, tidak ada jaminan bahwa instans baru yang diluncurkan dengan grup peluncuran yang sama juga akan terpenuhi. Contoh cara mengatur Grup Peluncuran ditampilkan dalam contoh kode berikut.


```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 5 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(5));

// Set the launch group.
requestRequest.setLaunchGroup("ADVANCED-DEMO-LAUNCH-GROUP");

// Set up the specifications of the launch. This includes
// the instance type (e.g. t1.micro) and the latest Amazon Linux
// AMI id available. Note, you should always use the latest
// Amazon Linux AMI id or another of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Jika Anda ingin memastikan bahwa semua instans dalam permintaan diluncurkan di Availability Zone yang sama, dan Anda tidak peduli yang mana, Anda dapat memanfaatkan grup Availability Zone. Grup Availability Zone adalah label yang mengelompokkan satu set instance bersama-sama di Availability Zone yang sama. Semua instance yang berbagi grup Availability Zone dan terpenuhi pada saat yang sama akan dimulai di Availability Zone yang sama. Contoh cara mengatur grup Availability Zone berikut.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();
```

```
// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 5 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(5));

// Set the availability zone group.
requestRequest.setAvailabilityZoneGroup("ADVANCED-DEMO-AZ-GROUP");

// Set up the specifications of the launch. This includes the instance
// type (e.g. t1.micro) and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon Linux AMI id or another
// of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Anda dapat menentukan Availability Zone yang Anda inginkan untuk Instans Spot. Contoh kode berikut menunjukkan cara mengatur Availability Zone.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));
```

```
// Set up the specifications of the launch. This includes the instance
// type (e.g. t1.micro) and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon Linux AMI id or another
// of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Set up the availability zone to use. Note we could retrieve the
// availability zones using the ec2.describeAvailabilityZones() API. For
// this demo we will just use us-east-1a.
SpotPlacement placement = new SpotPlacement("us-east-1b");
launchSpecification.setPlacement(placement);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Terakhir, Anda dapat menentukan grup penempatan jika Anda menggunakan Instans Spot Komputasi Kinerja Tinggi (HPC), seperti instans komputasi kluster atau instans GPU kluster. Grup penempatan memberi Anda latensi yang lebih rendah dan konektivitas bandwidth tinggi antara instans. Contoh cara mengatur grup penempatan berikut.

```
// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set up the specifications of the launch. This includes the instance
```

```
// type (e.g. t1.micro) and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon Linux AMI id or another
// of your choosing.

LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);

// Set up the placement group to use with whatever name you desire.
// For this demo we will just use "ADVANCED-DEMO-PLACEMENT-GROUP".
SpotPlacement placement = new SpotPlacement();
placement.setGroupName("ADVANCED-DEMO-PLACEMENT-GROUP");
launchSpecification.setPlacement(placement);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Semua parameter yang ditunjukkan dalam bagian ini adalah opsional. Penting juga untuk menyadari bahwa sebagian besar parameter ini—dengan pengecualian apakah tawaran Anda satu kali atau persisten—dapat mengurangi kemungkinan pemenuhan tawaran. Jadi, penting untuk memanfaatkan opsi ini hanya jika Anda membutuhkannya. Semua contoh kode sebelumnya digabungkan menjadi satu contoh kode panjang, yang dapat ditemukan di `com.amazonaws.codesamples.advanced.InlineGettingStartedCodeSampleApp.javakelas`.

Cara bertahan partisi root setelah gangguan atau penghentian

Salah satu cara termudah untuk mengelola gangguan Instans Spot Anda adalah memastikan bahwa data Anda diperiksa ke Amazon Elastic Block Store (Amazon EBS) volume pada irama biasa. Dengan checkpointing secara berkala, jika ada gangguan Anda akan kehilangan hanya data yang dibuat sejak pos pemeriksaan terakhir (dengan asumsi tidak ada tindakan non-idempoten lainnya yang dilakukan di antaranya). Agar proses ini lebih mudah, Anda dapat mengonfigurasi Permintaan Spot untuk memastikan bahwa partisi root Anda tidak akan dihapus saat gangguan

atau penghentian. Kami telah memasukkan kode baru dalam contoh berikut yang menunjukkan bagaimana mengaktifkan skenario ini.

Dalam kode tambahan, kita membuat `BlockDeviceMapping` objek dan mengatur terkait Amazon Elastic Block Store (Amazon EBS) ke sebuah Amazon EBS objek yang telah kita konfigurasi untuk tidak dihapus jika Instans Spot dihentikan. Kami kemudian menambahkan ini `BlockDeviceMapping` ke `ArrayList` pemetaan yang kami sertakan dalam spesifikasi peluncuran.

```
// Retrieves the credentials from an AWSCredentials.properties file.
AWSCredentials credentials = null;
try {
    credentials = new PropertiesCredentials(
        GettingStartedApp.class.getResourceAsStream("AwsCredentials.properties"));
}
catch (IOException e1) {
    System.out.println(
        "Credentials were not properly entered into AwsCredentials.properties.");
    System.out.println(e1.getMessage());
    System.exit(-1);
}

// Create the AmazonEC2 client so we can call various APIs.
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

// Initializes a Spot Instance Request
RequestSpotInstancesRequest requestRequest = new RequestSpotInstancesRequest();

// Request 1 x t1.micro instance with a bid price of $0.03.
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

// Set up the specifications of the launch. This includes the instance
// type (e.g. t1.micro) and the latest Amazon Linux AMI id available.
// Note, you should always use the latest Amazon Linux AMI id or another
// of your choosing.
LaunchSpecification launchSpecification = new LaunchSpecification();
launchSpecification.setImageId("ami-a9d09ed1");
launchSpecification.setInstanceType(InstanceType.T1Micro);

// Add the security group to the request.
ArrayList<String> securityGroups = new ArrayList<String>();
securityGroups.add("GettingStartedGroup");
launchSpecification.setSecurityGroups(securityGroups);
```

```
// Create the block device mapping to describe the root partition.
BlockDeviceMapping blockDeviceMapping = new BlockDeviceMapping();
blockDeviceMapping.setDeviceName("/dev/sda1");

// Set the delete on termination flag to false.
EbsBlockDevice ebs = new EbsBlockDevice();
ebs.setDeleteOnTermination(Boolean.FALSE);
blockDeviceMapping.setEbs(ebs);

// Add the block device mapping to the block list.
ArrayList<BlockDeviceMapping> blockList = new ArrayList<BlockDeviceMapping>();
blockList.add(blockDeviceMapping);

// Set the block device mapping configuration in the launch specifications.
launchSpecification.setBlockDeviceMappings(blockList);

// Add the launch specification.
requestRequest.setLaunchSpecification(launchSpecification);

// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult =
    ec2.requestSpotInstances(requestRequest);
```

Dengan asumsi Anda ingin melampirkan kembali volume ini ke instans Anda saat startup, Anda juga dapat menggunakan pengaturan pemetaan perangkat blok. Atau, jika Anda memasang partisi non-root, Anda dapat menentukan Amazon Amazon EBS volume yang ingin Anda lampirkan ke Instans Spot setelah dilanjutkan. Anda melakukan ini hanya dengan menentukan ID snapshot di `AmazonEbsBlockDevice` dan nama perangkat alternatif di `BlockDeviceMapping` objek. Dengan memanfaatkan pemetaan perangkat blok, dapat lebih mudah untuk bootstrap contoh Anda.

Menggunakan partisi root untuk memeriksa data kritis Anda adalah cara yang bagus untuk mengelola potensi gangguan instans Anda. Untuk metode lebih lanjut tentang mengelola potensi gangguan, silakan kunjungi [Mengelola Interupsi](#) video.

Cara menandai permintaan dan instans spot Anda

Menambahkan tanda ke Amazon EC2 sumber daya dapat menyederhanakan administrasi infrastruktur cloud Anda. Sebuah bentuk metadata, tag dapat digunakan untuk membuat nama yang user-friendly, meningkatkan penelusuran, dan meningkatkan koordinasi antara beberapa pengguna. Anda juga dapat menggunakan tanda untuk mengotomatisasi skrip dan bagian proses Anda. Untuk

membaca lebih lanjut tentang penandaan Amazon EC2 sumber daya, pergi ke [Menggunakan Tag](#) di dalam Amazon EC2 Panduan Pengguna untuk Instans Linux.

Permintaan penandaan

Untuk menambahkan tag ke permintaan tempat Anda, Anda perlu memberi tag pada mereka setelah mereka telah diminta. Nilai kembali dari `requestSpotInstances()` menyediakan Anda dengan [RequestSpotInstancesResult](#) objek yang dapat Anda gunakan untuk mendapatkan ID permintaan spot untuk penandaan:

```
// Call the RequestSpotInstance API.
RequestSpotInstancesResult requestResult = ec2.requestSpotInstances(requestRequest);
List<SpotInstanceRequest> requestResponses = requestResult.getSpotInstanceRequests();

// A list of request IDs to tag
ArrayList<String> spotInstanceRequestIds = new ArrayList<String>();

// Add the request ids to the hashset, so we can determine when they hit the
// active state.
for (SpotInstanceRequest requestResponse : requestResponses) {
    System.out.println("Created Spot Request:
"+requestResponse.getSpotInstanceRequestId());
    spotInstanceRequestIds.add(requestResponse.getSpotInstanceRequestId());
}
```

Setelah Anda memiliki ID, Anda dapat menandai permintaan dengan menambahkan ID mereka ke [CreateTagsRequest](#) dan memanggil `Amazon EC2 klien createTags()` Metode:

```
// The list of tags to create
ArrayList<Tag> requestTags = new ArrayList<Tag>();
requestTags.add(new Tag("keyname1", "value1"));

// Create the tag request
CreateTagsRequest createTagsRequest_requests = new CreateTagsRequest();
createTagsRequest_requests.setResources(spotInstanceRequestIds);
createTagsRequest_requests.setTags(requestTags);

// Tag the spot request
try {
    ec2.createTags(createTagsRequest_requests);
}
catch (AmazonServiceException e) {
```

```

System.out.println("Error terminating instances");
System.out.println("Caught Exception: " + e.getMessage());
System.out.println("Reponse Status Code: " + e.getStatusCode());
System.out.println("Error Code: " + e.getErrorCode());
System.out.println("Request ID: " + e.getRequestId());
}

```

Menandai instans

Demikian pula untuk melihat permintaan sendiri, Anda hanya dapat menandai sebuah instance setelah dibuat, yang akan terjadi setelah permintaan spot telah terpenuhi (itu tidak lagi diterbukanegara).

Anda dapat memeriksa status permintaan Anda dengan menghubungi Amazon EC2 `describeSpotInstanceRequests()` metode dengan [DescribeSpotInstanceRequestsRequest](#) objek.

Kembali [DescribeSpotInstanceRequestsResult](#) objek berisi daftar [SpotInstanceRequest](#) objek yang dapat Anda gunakan untuk query status permintaan spot Anda dan mendapatkan ID instans mereka setelah mereka tidak lagi diterbukanegara.

Setelah permintaan spot tidak lagi terbuka, Anda dapat mengambil ID instans dari [SpotInstanceRequest](#) objek dengan memanggil `getInstanceId()` metode.

```

boolean anyOpen; // tracks whether any requests are still open

// a list of instances to tag.
ArrayList<String> instanceIds = new ArrayList<String>();

do {
    DescribeSpotInstanceRequestsRequest describeRequest =
        new DescribeSpotInstanceRequestsRequest();
    describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

    anyOpen=false; // assume no requests are still open

    try {
        // Get the requests to monitor
        DescribeSpotInstanceRequestsResult describeResult =
            ec2.describeSpotInstanceRequests(describeRequest);

        List<SpotInstanceRequest> describeResponses =
            describeResult.getSpotInstanceRequests();
    }
}

```



```
// are any requests open?
for (SpotInstanceRequest describeResponse : describeResponses) {
    if (describeResponse.getState().equals("open")) {
        anyOpen = true;
        break;
    }
    // get the corresponding instance ID of the spot request
    instanceIds.add(describeResponse.getInstanceId());
}
}
catch (AmazonServiceException e) {
    // Don't break the loop due to an exception (it may be a temporary issue)
    anyOpen = true;
}

try {
    Thread.sleep(60*1000); // sleep 60s.
}
catch (Exception e) {
    // Do nothing if the thread woke up early.
}
} while (anyOpen);
```

Sekarang Anda dapat menandai contoh yang dikembalikan:

```
// Create a list of tags to create
ArrayList<Tag> instanceTags = new ArrayList<Tag>();
instanceTags.add(new Tag("keyname1","value1"));

// Create the tag request
CreateTagsRequest createTagsRequest_instances = new CreateTagsRequest();
createTagsRequest_instances.setResources(instanceIds);
createTagsRequest_instances.setTags(instanceTags);

// Tag the instance
try {
    ec2.createTags(createTagsRequest_instances);
}
catch (AmazonServiceException e) {
    // Write out any exceptions that may have occurred.
    System.out.println("Error terminating instances");
    System.out.println("Caught Exception: " + e.getMessage());
}
```

```
System.out.println("Reponse Status Code: " + e.getStatusCode());
System.out.println("Error Code: " + e.getErrorCode());
System.out.println("Request ID: " + e.getRequestId());
}
```

Membatalkan permintaan spot dan mengakhiri instans

Membatalkan permintaan spot

Untuk membatalkan permintaan Instans Spot Instans Spot
InstanscancelSpotInstanceRequestspadaAmazon EC2klien
dengan[CancelSpotInstanceRequestsRequest](#)objek.

```
try {
    CancelSpotInstanceRequestsRequest cancelRequest = new
    CancelSpotInstanceRequestsRequest(spotInstanceRequestIds);
    ec2.cancelSpotInstanceRequests(cancelRequest);
} catch (AmazonServiceException e) {
    System.out.println("Error cancelling instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

Mengakhiri Instans Spot

Anda dapat mengakhiri Instans Spot apa pun yang berjalan dengan meneruskan ID mereka
keAmazon EC2klienterminateInstances()metode.

```
try {
    TerminateInstancesRequest terminateRequest = new
    TerminateInstancesRequest(instanceIds);
    ec2.terminateInstances(terminateRequest);
} catch (AmazonServiceException e) {
    System.out.println("Error terminating instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

Membawa semuanya bersama-sama

Untuk menyatukan semua ini, kami menyediakan pendekatan yang lebih berorientasi objek yang menggabungkan langkah-langkah yang kami tunjukkan dalam tutorial ini menjadi satu kelas yang mudah digunakan. Kami instantiate kelas disebut `Request` yang melakukan tindakan ini. Kami juga membuat `GettingStartedApp` kelas, yang memiliki metode utama di mana kita melakukan panggilan fungsi tingkat tinggi.

Kode sumber lengkap untuk contoh ini dapat dilihat atau diunduh di [GitHub](#).

Selamat! Anda telah menyelesaikan tutorial Fitur Permintaan Lanjutan untuk mengembangkan perangkat lunak Instans Spot dengan AWS SDK for Java.

Mengelola Amazon EC2 Instans

Membuat Instans

Membuat Amazon EC2 contoh dengan memanggil `AmazonEC2Client.runInstances` metode, menyediakannya dengan [RunInstancesRequest](#) mengandung [Amazon Machine Image \(AMI\)](#) untuk menggunakan dan [type instans](#).

Impor

```
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.InstanceType;
import com.amazonaws.services.ec2.model.RunInstancesRequest;
import com.amazonaws.services.ec2.model.RunInstancesResult;
import com.amazonaws.services.ec2.model.Tag;
```

Kode

```
RunInstancesRequest run_request = new RunInstancesRequest()
    .withImageId(ami_id)
    .withInstanceType(InstanceType.T1Micro)
    .withMaxCount(1)
    .withMinCount(1);

RunInstancesResult run_response = ec2.runInstances(run_request);

String reservation_id =
    run_response.getReservation().getInstances().get(0).getInstanceId();
```

Lihat [Lengkapi Contoh](#).

Memulai Instans

Untuk memulai Amazon EC2 misalnya, hubungi `AmazonEC2Client.startInstances` metode, menyediakannya dengan [StartInstancesRequest](#) berisi ID instance untuk memulai.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.StartInstancesRequest;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

StartInstancesRequest request = new StartInstancesRequest()
    .withInstanceIds(instance_id);

ec2.startInstances(request);
```

Lihat [Lengkapi Contoh](#).

Menghentikan Instans

Untuk menghentikan Amazon EC2 misalnya, hubungi `AmazonEC2Client.stopInstances` metode, menyediakannya dengan [StopInstancesRequest](#) berisi ID instance untuk berhenti.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.StopInstancesRequest;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

StopInstancesRequest request = new StopInstancesRequest()
    .withInstanceIds(instance_id);
```

```
ec2.stopInstances(request);
```

Lihat [Lengkapi Contoh](#).

Mem-boot Ulang Instans

Untuk me-reboot Amazon EC2 misalnya, hubungi `AmazonEC2Client.rebootInstances` metode, menyediakannya dengan [RebootInstancesRequest](#) berisi ID instance untuk reboot.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.RebootInstancesRequest;
import com.amazonaws.services.ec2.model.RebootInstancesResult;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

RebootInstancesRequest request = new RebootInstancesRequest()
    .withInstanceIds(instance_id);

RebootInstancesResult response = ec2.rebootInstances(request);
```

Lihat [Lengkapi Contoh](#).

Menjelaskan Instans

Untuk mencantumkan instans Anda, buat [DescribeInstancesRequest](#) dan memanggil `AmazonEC2Client.describeInstances` metode. Ini akan mengembalikan [DescribeInstancesResult](#) objek yang dapat Anda gunakan untuk daftar Amazon EC2 instans untuk akun dan wilayah Anda.

Instans dikelompokkan berdasarkan kekhawatiran. Setiap reservasi sesuai dengan panggilan `startInstances` yang meluncurkan instance. Untuk mencantumkan instans Anda, Anda harus memanggil `DescribeInstancesResult` kelas `getReservations` method, and then call `getInstances` pada setiap kembali [Reservasi](#) objek.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
```

```
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
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.Reservation;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();
boolean done = false;

DescribeInstancesRequest request = new DescribeInstancesRequest();
while(!done) {
    DescribeInstancesResult response = ec2.describeInstances(request);

    for(Reservation reservation : response.getReservations()) {
        for(Instance instance : reservation.getInstances()) {
            System.out.printf(
                "Found instance with id %s, " +
                "AMI %s, " +
                "type %s, " +
                "state %s " +
                "and monitoring state %s",
                instance.getInstanceId(),
                instance.getImageId(),
                instance.getInstanceType(),
                instance.getState().getName(),
                instance.getMonitoring().getState());
        }
    }

    request.setNextToken(response.getNextToken());

    if(response.getNextToken() == null) {
        done = true;
    }
}
```

Hasil yang paged; Anda bisa mendapatkan hasil lebih lanjut dengan melewati nilai yang dikembalikan dari objek `getNextToken` metode untuk objek permintaan asli `Anda.setNextToken` metode, kemudian menggunakan objek permintaan yang sama dalam panggilan berikutnya untuk `describeInstances`.

Lihat [Lengkapi Contoh](#).

Memantau Instans

Anda dapat memantau berbagai aspek Amazon EC2 contoh, seperti CPU dan pemanfaatan jaringan, memori yang tersedia, dan ruang disk yang tersisa. Untuk mempelajari tentang pemantauan instans, lihat [Pemantauan Amazon EC2](#) di Amazon EC2 Panduan Pengguna Instans Linux.

Untuk memulai pemantauan sebuah instans, Anda harus membuat [MonitorInstancesRequest](#) dengan ID instance untuk memantau, dan menyebarkannya ke `AmazonEC2Client.monitorInstances` metode.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.MonitorInstancesRequest;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

MonitorInstancesRequest request = new MonitorInstancesRequest()
    .withInstanceIds(instance_id);

ec2.monitorInstances(request);
```

Lihat [Lengkapi Contoh](#).

Memantau Instans

Untuk menghentikan pemantauan instance, buat [UnmonitorInstancesRequest](#) dengan ID instance untuk menghentikan pemantauan, dan menyebarkannya ke `AmazonEC2Client.unmonitorInstances` metode.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.UnmonitorInstancesRequest;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

UnmonitorInstancesRequest request = new UnmonitorInstancesRequest()
    .withInstanceIds(instance_id);

ec2.unmonitorInstances(request);
```

Lihat [Lengkapi Contoh](#).

Informasi Selengkapnya

- [RunInstances](#) di Amazon EC2 Referensi API
- [DescribeInstances](#) di Amazon EC2 Referensi API
- [StartInstances](#) di Amazon EC2 Referensi API
- [StopInstances](#) di Amazon EC2 Referensi API
- [RebootInstances](#) di Amazon EC2 Referensi API
- [MonitorInstances](#) di Amazon EC2 Referensi API
- [UnmonitorInstances](#) di Amazon EC2 Referensi API

Menggunakan Alamat IP Elastis di Amazon EC2

EC2-Classic akan pensiun

Warning

Kami pensiun EC2-Classic pada 15 Agustus 2022. Kami menyarankan Anda bermigrasi dari EC2-Classic ke VPC. [Untuk informasi selengkapnya, lihat Memigrasi dari EC2-Classic ke VPC di Panduan Pengguna Amazon EC2 atau Panduan Pengguna Amazon EC2](#). Lihat juga posting blog [EC2-Classic-Classic Networking is Retiring - Inilah](#) Cara Mempersiapkan.

Mengalokasikan Alamat IP Elastis

Untuk menggunakan alamat IP Elastis, pertama-tama Anda mengalokasikannya ke akun Anda, lalu mengaitkannya dengan instans atau antarmuka jaringan.

Untuk mengalokasikan alamat IP Elastis, panggil `allocateAddress` metode `AmazonEC2Client` dengan [AllocateAddressRequest](#) objek yang berisi tipe jaringan (EC2 klasik atau VPC).

Yang dikembalikan [AllocateAddressResult](#) berisi ID alokasi yang dapat Anda gunakan untuk mengaitkan alamat dengan instance, dengan meneruskan ID alokasi dan ID instance [AssociateAddressRequest](#) ke metode AmazonEC2Client. `associateAddress`

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.AllocateAddressRequest;
import com.amazonaws.services.ec2.model.AllocateAddressResult;
import com.amazonaws.services.ec2.model.AssociateAddressRequest;
import com.amazonaws.services.ec2.model.AssociateAddressResult;
import com.amazonaws.services.ec2.model.DomainType;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

AllocateAddressRequest allocate_request = new AllocateAddressRequest()
    .withDomain(DomainType.Vpc);

AllocateAddressResult allocate_response =
    ec2.allocateAddress(allocate_request);

String allocation_id = allocate_response.getAllocationId();

AssociateAddressRequest associate_request =
    new AssociateAddressRequest()
        .withInstanceId(instance_id)
        .withAllocationId(allocation_id);

AssociateAddressResult associate_response =
    ec2.associateAddress(associate_request);
```

Lihat [contoh lengkapnya](#).

Menjelaskan Alamat IP Elastis

Untuk membuat daftar alamat IP Elastis yang ditetapkan ke akun Anda, hubungi metode AmazonEC2Client. `describeAddresses` Ini mengembalikan [DescribeAddressesResult](#) yang dapat Anda gunakan untuk mendapatkan daftar objek [Alamat](#) yang mewakili alamat IP Elastis di akun Anda.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.Address;
import com.amazonaws.services.ec2.model.DescribeAddressesResult;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

DescribeAddressesResult response = ec2.describeAddresses();

for(Address address : response.getAddresses()) {
    System.out.printf(
        "Found address with public IP %s, " +
        "domain %s, " +
        "allocation id %s " +
        "and NIC id %s",
        address.getPublicIp(),
        address.getDomain(),
        address.getAllocationId(),
        address.getNetworkInterfaceId());
}
```

Lihat [contoh lengkapnya](#).

Melepaskan Alamat IP Elastis

Untuk merilis alamat IP Elastis, panggil `releaseAddress` metode `AmazonEC2Client`, berikan ID alokasi yang [ReleaseAddressRequest](#) berisi ID alokasi alamat IP Elastis yang ingin Anda lepaskan.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.ReleaseAddressRequest;
import com.amazonaws.services.ec2.model.ReleaseAddressResult;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();
```

```
ReleaseAddressRequest request = new ReleaseAddressRequest()
    .withAllocationId(alloc_id);

ReleaseAddressResult response = ec2.releaseAddress(request);
```

Setelah Anda merilis alamat IP Elastis, itu dilepaskan ke kumpulan alamat AWS IP dan mungkin tidak tersedia untuk Anda sesudahnya. Pastikan untuk memperbarui catatan DNS Anda dan server atau perangkat apa pun yang berkomunikasi dengan alamat tersebut. Jika Anda mencoba melepaskan alamat IP Elastis yang sudah dirilis, Anda akan mendapatkan `AuthFailure` kesalahan jika alamat tersebut sudah dialokasikan ke alamat lain Akun AWS.

Jika Anda menggunakan EC2-Classic atau VPC default, maka melepaskan alamat IP Elastis secara otomatis memisahkannya dari instance apa pun yang terkait dengannya. Untuk memisahkan alamat IP Elastis tanpa melepaskannya, gunakan metode `AmazonEC2Client.disassociateAddress`

Jika Anda menggunakan VPC non-default, Anda harus `disassociateAddress` menggunakan untuk memisahkan alamat IP Elastic sebelum Anda mencoba melepaskannya. Jika tidak, Amazon EC2 mengembalikan kesalahan (`InvalidIpAddress.InUse`).

Lihat [contoh lengkapnya](#).

Informasi Selengkapnya

- [Alamat IP Elastis](#) dalam Panduan Amazon EC2 Pengguna untuk Instans Linux
- [AllocateAddress](#) di Referensi Amazon EC2 API
- [DescribeAddresses](#) di Referensi Amazon EC2 API
- [ReleaseAddress](#) di Referensi Amazon EC2 API

Gunakan wilayah dan availability zone

Mendeskripsikan wilayah

Untuk mencantumkan Wilayah yang tersedia untuk akun Anda, hubungi `AmazonEC2Client.describeRegions` metode. Ini menghasilkan [DescribeRegionsResult](#). Panggil objek yang dikembalikan `getRegions` metode untuk mendapatkan daftar [Wilayah](#) benda-benda yang mewakili setiap Daerah.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeRegionsResult;
import com.amazonaws.services.ec2.model.Region;
import com.amazonaws.services.ec2.model.AvailabilityZone;
import com.amazonaws.services.ec2.model.DescribeAvailabilityZonesResult;
```

Kode

```
DescribeRegionsResult regions_response = ec2.describeRegions();

for(Region region : regions_response.getRegions()) {
    System.out.printf(
        "Found region %s " +
        "with endpoint %s",
        region.getRegionName(),
        region.getEndpoint());
}
```

Lihat [Lengkapi Contoh](#).

Mendesripsikan zona ketersediaan

Untuk mencantumkan setiap Availability Zone yang tersedia untuk akun Anda, hubungi `AmazonEC2Client.describeAvailabilityZones` metode. Ini menghasilkan [DescribeAvailabilityZonesResult](#). Memanggil `getAvailabilityZones` metode untuk mendapatkan daftar [AvailabilityZone](#) objek yang mewakili setiap Availability Zone.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeRegionsResult;
import com.amazonaws.services.ec2.model.Region;
import com.amazonaws.services.ec2.model.AvailabilityZone;
import com.amazonaws.services.ec2.model.DescribeAvailabilityZonesResult;
```

Kode

```
DescribeAvailabilityZonesResult zones_response =
    ec2.describeAvailabilityZones();
```

```
for(AvailabilityZone zone : zones_response.getAvailabilityZones()) {
    System.out.printf(
        "Found availability zone %s " +
        "with status %s " +
        "in region %s",
        zone.getZoneName(),
        zone.getState(),
        zone.getRegionName());
}
```

Lihat [Lengkapi Contoh](#).

Mendeskripsikan akun

Untuk menggambarkan akun Anda, hubungi `AmazonEC2Client.describeAccountAttributes` metode. Metode ini mengembalikan [DescribeAccountAttributesResult](#) objek. Memanggil objek `in.getAccountAttributes` metode untuk mendapatkan daftar [AccountAttribute](#) objek. Anda dapat iterate melalui daftar untuk mengambil [AccountAttribute](#) objek.

Anda bisa mendapatkan nilai atribut akun Anda dengan menerapkan [AccountAttribute](#) objek `getAttributeValues` metode. Metode ini menghasilkan daftar [AccountAttributeValue](#) objek. Anda dapat iterate melalui daftar kedua ini untuk menampilkan nilai atribut (lihat contoh kode berikut).

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.AccountAttributeValue;
import com.amazonaws.services.ec2.model.DescribeAccountAttributesResult;
import com.amazonaws.services.ec2.model.AccountAttribute;
import java.util.List;
import java.util.ListIterator;
```

Kode

```
AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

try{
    DescribeAccountAttributesResult accountResults = ec2.describeAccountAttributes();
```

```
List<AccountAttribute> accountList = accountResults.getAccountAttributes();

for (ListIterator iter = accountList.listIterator(); iter.hasNext(); ) {

    AccountAttribute attribute = (AccountAttribute) iter.next();
    System.out.print("\n The name of the attribute is
"+attribute.getAttributeName());
    List<AccountAttributeValue> values = attribute.getAttributeValues();

    //iterate through the attribute values
    for (ListIterator iterVals = values.listIterator(); iterVals.hasNext(); ) {
        AccountAttributeValue myValue = (AccountAttributeValue) iterVals.next();
        System.out.print("\n The value of the attribute is
"+myValue.getAttributeValue());
    }
}
System.out.print("Done");
}
catch (Exception e)
{
    e.printStackTrace();
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Informasi lain

- [Wilayah dan Availability Zone](#) di dalam Amazon EC2 Panduan Pengguna untuk Instans Linux
- [DescribeRegions](#) di dalam Amazon EC2 Referensi API
- [DescribeAvailabilityZones](#) di dalam Amazon EC2 Referensi API

Bekerja dengan Amazon EC2 Pasangan Kunci

Membuat Pasangan Kunci

Untuk membuat key pair, panggil `AmazonEC2Client.createKeyPair` metode dengan [CreateKeyPairRequest](#) yang berisi nama kunci.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
```

```
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.CreateKeyPairRequest;
import com.amazonaws.services.ec2.model.CreateKeyPairResult;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

CreateKeyPairRequest request = new CreateKeyPairRequest()
    .withKeyName(key_name);

CreateKeyPairResult response = ec2.createKeyPair(request);
```

Lihat [Lengkapi Contoh](#).

Menggambarkan Pasangan Kunci

Untuk mencantumkan pasangan kunci Anda atau untuk mendapatkan informasi tentang mereka, hubungi `AmazonEC2Client.describeKeyPairs` metode. Ini mengembalikan [DescribeKeyPairsResult](#) yang dapat Anda gunakan untuk mengakses daftar pasangan kunci dengan memanggil `getKeyPairs` metode, yang mengembalikan daftar [KeyPairInfo](#) objek.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeKeyPairsResult;
import com.amazonaws.services.ec2.model.KeyPairInfo;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

DescribeKeyPairsResult response = ec2.describeKeyPairs();

for(KeyPairInfo key_pair : response.getKeyPairs()) {
    System.out.printf(
        "Found key pair with name %s " +
        "and fingerprint %s",
        key_pair.getKeyName(),
        key_pair.getKeyFingerprint());
}
```

```
}
```

Lihat [Lengkapi Contoh](#).

Menghapus Pasangan Kunci

Untuk menghapus key pair, panggil `AmazonEC2Client.deleteKeyPair` metode, melewatinya [DeleteKeyPairRequest](#) yang berisi nama key pair.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DeleteKeyPairRequest;
import com.amazonaws.services.ec2.model.DeleteKeyPairResult;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

DeleteKeyPairRequest request = new DeleteKeyPairRequest()
    .withKeyName(key_name);

DeleteKeyPairResult response = ec2.deleteKeyPair(request);
```

Lihat [Lengkapi Contoh](#).

Informasi Selengkapnya

- [Amazon EC2 Pasangan Kunci](#) di Amazon EC2 Panduan Pengguna untuk Instans Linux
- [CreateKeyPair](#) di Amazon EC2 Referensi API
- [DescribeKeyPairs](#) di Amazon EC2 Referensi API
- [DeleteKeyPair](#) di Amazon EC2 Referensi API

Bekerja dengan Grup Amazon EC2

Membuat Grup Keamanan

Untuk membuat grup keamanan, panggil `createSecurityGroup` metode `AmazonEC2Client` dengan [CreateSecurityGroupRequest](#) yang berisi nama kunci.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.CreateSecurityGroupRequest;
import com.amazonaws.services.ec2.model.CreateSecurityGroupResult;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

CreateSecurityGroupRequest create_request = new
    CreateSecurityGroupRequest()
        .withGroupName(group_name)
        .withDescription(group_desc)
        .withVpcId(vpc_id);

CreateSecurityGroupResult create_response =
    ec2.createSecurityGroup(create_request);
```

Lihat [contoh lengkapnya](#).

Mengonfigurasi sebuah grup

Grup keamanan dapat mengontrol lalu lintas masuk (ingress) dan outbound (egress) ke Amazon EC2 instans Anda.

Untuk menambahkan aturan ingress ke grup keamanan Anda, gunakan `authorizeSecurityGroupIngress` metode `AmazonEC2Client`, dengan memberikan nama grup keamanan dan aturan akses ([IPPermission](#)) yang ingin Anda tetapkan di dalam [AuthorizeSecurityGroupIngressRequest](#) objek. Contoh berikut menunjukkan cara menambahkan izin ke sebuah grup.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.CreateSecurityGroupRequest;
import com.amazonaws.services.ec2.model.CreateSecurityGroupResult;
```

Kode

```
IpRange ip_range = new IpRange()
    .withCidrIp("0.0.0.0/0");

IpPermission ip_perm = new IpPermission()
    .withIpProtocol("tcp")
    .withToPort(80)
    .withFromPort(80)
    .withIpv4Ranges(ip_range);

IpPermission ip_perm2 = new IpPermission()
    .withIpProtocol("tcp")
    .withToPort(22)
    .withFromPort(22)
    .withIpv4Ranges(ip_range);

AuthorizeSecurityGroupIngressRequest auth_request = new
    AuthorizeSecurityGroupIngressRequest()
        .withGroupName(group_name)
        .withIpPermissions(ip_perm, ip_perm2);

AuthorizeSecurityGroupIngressResult auth_response =
    ec2.authorizeSecurityGroupIngress(auth_request);
```

Untuk menambahkan aturan egress ke grup keamanan, berikan data serupa [AuthorizeSecurityGroupEgressRequest](#) ke `authorizeSecurityGroupEgress` metode `AmazonEC2Client`.

Lihat [contoh lengkapnya](#).

Menggambarkan Grup Keamanan

Untuk menggambarkan grup keamanan Anda atau mendapatkan informasi tentang mereka, hubungi `describeSecurityGroups` metode `AmazonEC2Client`. Ia mengembalikan [DescribeSecurityGroupsResult](#) yang dapat Anda gunakan untuk mengakses daftar kelompok keamanan dengan memanggil `getSecurityGroups` metodenya, yang mengembalikan daftar [SecurityGroup](#) objek.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
```

```
import com.amazonaws.services.ec2.model.DescribeSecurityGroupsRequest;
import com.amazonaws.services.ec2.model.DescribeSecurityGroupsResult;
```

Kode

```
final String USAGE =
    "To run this example, supply a group id\n" +
    "Ex: DescribeSecurityGroups <group-id>\n";

if (args.length != 1) {
    System.out.println(USAGE);
    System.exit(1);
}

String group_id = args[0];
```

Lihat [contoh lengkapnya](#).

Menghapus sebuah grup

Untuk menghapus grup keamanan, panggil `deleteSecurityGroup` metode `AmazonEC2Client`, meneruskannya [DeleteSecurityGroupRequest](#) yang berisi ID grup keamanan untuk dihapus.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DeleteSecurityGroupRequest;
import com.amazonaws.services.ec2.model.DeleteSecurityGroupResult;
```

Kode

```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();

DeleteSecurityGroupRequest request = new DeleteSecurityGroupRequest()
    .withGroupId(group_id);

DeleteSecurityGroupResult response = ec2.deleteSecurityGroup(request);
```

Lihat [contoh lengkapnya](#).

Informasi Selengkapnya

- [Amazon EC2Grup Keamanan](#) di PanduanAmazon EC2 Pengguna untuk Instans Linux
- [Otorisasi Lalu Lintas Masuk untuk Instans Linux Anda](#) diAmazon EC2 User Guide untuk Instans Linux
- [CreateSecurityGroup](#)di ReferensiAmazon EC2 API
- [DescribeSecurityGroups](#)di ReferensiAmazon EC2 API
- [DeleteSecurityGroup](#)di ReferensiAmazon EC2 API
- [AuthorizeSecurityGroupIngress](#)di ReferensiAmazon EC2 API

Contoh IAM MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrogramanIAMdengan menggunakan[AWS SDK for Java](#).

AWS Identity and Access Management(IAM) memungkinkan Anda untuk mengontrol akses keAWSSlayanan dan sumber daya untuk pengguna Anda. Menggunakan IAM, Anda dapat membuat dan mengelolaAWSpengguna dan grup, dan menggunakan izin untuk mengizinkan dan menolak akses mereka keAWSSumber daya. Untuk panduan lengkap untuk IAM, kunjungi[IAMPanduan Pengguna](#).

Note

Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parameter[contoh kode tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk membangun dan menjalankan.

Topik

- [Mengelola Kunci Akses IAM](#)
- [Membuat pengguna IAM](#)
- [Menggunakan Alias Akun IAM](#)
- [Cara menggunakan kebijakan IAM](#)
- [Cara menggunakan Sertifikat Server IAM](#)

Mengelola Kunci Akses IAM

Membuat Kunci Akses

Untuk membuat kunci akses IAM, panggil `AmazonIdentityManagementClient.createAccessKeyMetode` dengan [CreateAccessKeyRequest](#) objek.

`CreateAccessKeyRequest` memiliki dua konstruktor - satu yang mengambil nama pengguna dan satu lagi tanpa parameter. Jika Anda menggunakan versi yang tidak mengambil parameter, Anda harus mengatur nama pengguna menggunakan `withUserName` metode setter sebelum meneruskannya ke `createAccessKeyMetode`.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateAccessKeyRequest;
import com.amazonaws.services.identitymanagement.model.CreateAccessKeyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

CreateAccessKeyRequest request = new CreateAccessKeyRequest()
    .withUserName(user);

CreateAccessKeyResult response = iam.createAccessKey(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Kunci akses

Untuk mencantumkan kunci akses untuk pengguna tertentu, buat [ListAccessKeysRequest](#) objek yang berisi nama pengguna untuk daftar kunci untuk, dan meneruskannya ke `AmazonIdentityManagementClient.listAccessKeysMetode`.

Note

Jika Anda tidak memberikan nama pengguna `kelistAccessKeys`, ia akan mencoba untuk daftar kunci akses yang terkait dengan Akun AWS yang menandatangani permintaan.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.AccessKeyMetadata;
import com.amazonaws.services.identitymanagement.model.ListAccessKeysRequest;
import com.amazonaws.services.identitymanagement.model.ListAccessKeysResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

boolean done = false;
ListAccessKeysRequest request = new ListAccessKeysRequest()
    .withUserName(username);

while (!done) {

    ListAccessKeysResult response = iam.listAccessKeys(request);

    for (AccessKeyMetadata metadata :
        response.getAccessKeyMetadata()) {
        System.out.format("Retrieved access key %s",
            metadata.getAccessKeyId());
    }

    request.setMarker(response.getMarker());

    if (!response.getIsTruncated()) {
        done = true;
    }
}
```

Hasil `listAccessKeys` yang paged (dengan maksimum default 100 catatan per panggilan). Anda dapat menelepon `getIsTruncated` pada dikembalikan `ListAccessKeysResult` objek untuk melihat apakah query kembali hasil yang lebih sedikit maka tersedia. Jika demikian, maka hubungi `setMarker` pada `ListAccessKeysRequest` dan menyebarkannya kembali ke doa berikutnya `listAccessKeys`.

Lihat [Lengkapi Contoh](#) di GitHub.

Mengambil Waktu Terakhir Digunakan Kunci Akses

Untuk mendapatkan waktu kunci akses terakhir digunakan, hubungi `AmazonIdentityManagementClient`'s `getAccessKeyLastUsed` metode dengan ID kunci akses (yang dapat diteruskan menggunakan `GetAccessKeyLastUsedRequest` objek, atau langsung ke overload yang mengambil kunci akses ID langsung.

Anda kemudian dapat menggunakan kembali `GetAccessKeyLastUsedResult` objek untuk mengambil kunci terakhir digunakan waktu.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetAccessKeyLastUsedRequest;
import com.amazonaws.services.identitymanagement.model.GetAccessKeyLastUsedResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

GetAccessKeyLastUsedRequest request = new GetAccessKeyLastUsedRequest()
    .withAccessKeyId(access_id);

GetAccessKeyLastUsedResult response = iam.getAccessKeyLastUsed(request);

System.out.println("Access key was last used at: " +
    response.getAccessKeyLastUsed().getLastUsedDate());
```

Lihat [Lengkapi Contoh](#) di GitHub.

Mengaktifkan atau Menonaktifkan Kunci Akses

Anda dapat mengaktifkan atau menonaktifkan kunci akses dengan membuat [UpdateAccessKeyRequest](#) objek, menyediakan ID kunci akses, opsional nama pengguna, dan yang diinginkan [Status](#), kemudian meneruskan objek permintaan ke `AmazonIdentityManagementClient.updateAccessKeyMetode`.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;  
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;  
import com.amazonaws.services.identitymanagement.model.UpdateAccessKeyRequest;  
import com.amazonaws.services.identitymanagement.model.UpdateAccessKeyResult;
```

Kode

```
final AmazonIdentityManagement iam =  
    AmazonIdentityManagementClientBuilder.defaultClient();  
  
UpdateAccessKeyRequest request = new UpdateAccessKeyRequest()  
    .withAccessKeyId(access_id)  
    .withUserName(username)  
    .withStatus(status);  
  
UpdateAccessKeyResult response = iam.updateAccessKey(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Menghapus Access key

Untuk menghapus kunci akses secara permanen, panggil `AmazonIdentityManagementClient.deleteKeymetode`, menyediakannya dengan [DeleteAccessKeyRequest](#) berisi ID dan nama pengguna kunci akses.

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 com.amazonaws.services.identitymanagement.AmazonIdentityManagement;  
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;  
import com.amazonaws.services.identitymanagement.model.DeleteAccessKeyRequest;  
import com.amazonaws.services.identitymanagement.model.DeleteAccessKeyResult;
```

Kode

```
final AmazonIdentityManagement iam =  
    AmazonIdentityManagementClientBuilder.defaultClient();  
  
DeleteAccessKeyRequest request = new DeleteAccessKeyRequest()  
    .withAccessKeyId(access_key)  
    .withUserName(username);  
  
DeleteAccessKeyResult response = iam.deleteAccessKey(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Informasi Selengkapnya

- [CreateAccessKey](#) dalam Referensi API IAM
- [ListAccessKeys](#) dalam Referensi API IAM
- [GetAccessKeyLastUsed](#) dalam Referensi API IAM
- [UpdateAccessKey](#) dalam Referensi API IAM
- [DeleteAccessKey](#) dalam Referensi API IAM

Membuat pengguna IAM

Membuat pengguna

Buat pengguna IAM baru dengan memberikan nama pengguna untuk `createUser` metode `AmazonIdentityManagementClient` ini, baik secara langsung atau menggunakan [CreateUserRequest](#) objek yang berisi nama pengguna.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
```

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateUserRequest;
import com.amazonaws.services.identitymanagement.model.CreateUserResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

CreateUserRequest request = new CreateUserRequest()
    .withUserName(username);

CreateUserResult response = iam.createUser(request);
```

Lihat [contoh lengkapnya](#) GitHub.

Daftar pengguna

Untuk mencantumkan pengguna IAM untuk akun Anda, buat yang baru [ListUsersRequest](#) dan teruskan ke `listUsers` metode tersebut `AmazonIdentityManagementClient`. Anda dapat mengambil daftar pengguna dengan memanggil `getUsers` [ListUsersResult](#) objek yang dikembalikan.

Daftar pengguna yang dikembalikan `listUsers` oleh `paged`. Anda dapat memeriksa untuk melihat ada lebih banyak hasil untuk diambil dengan memanggil `isTruncated` metode objek respon. Jika kembali `true`, kemudian memanggil `setMarker()` metode permintaan objek, lewat itu nilai kembali dari `getMarker()` metode respon objek.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListUsersRequest;
import com.amazonaws.services.identitymanagement.model.ListUsersResult;
import com.amazonaws.services.identitymanagement.model.User;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();
```

```
boolean done = false;
ListUsersRequest request = new ListUsersRequest();

while(!done) {
    ListUsersResult response = iam.listUsers(request);

    for(User user : response.getUsers()) {
        System.out.format("Retrieved user %s", user.getUserName());
    }

    request.setMarker(response.getMarker());

    if(!response.getIsTruncated()) {
        done = true;
    }
}
```

Lihat [contoh lengkapnya](#) GitHub.

Membuat pengguna

Untuk memperbarui pengguna, panggil `updateUser` metode `AmazonIdentityManagementClient` objek, yang mengambil [UpdateUserRequest](#) objek yang dapat Anda gunakan untuk mengubah nama atau jalur pengguna.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.UpdateUserRequest;
import com.amazonaws.services.identitymanagement.model.UpdateUserResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

UpdateUserRequest request = new UpdateUserRequest()
    .withUserName(cur_name)
    .withNewUserName(new_name);

UpdateUserResult response = iam.updateUser(request);
```

Lihat [contoh lengkapnya](#) GitHub.

Membuat pengguna

Untuk menghapus pengguna, panggil `AmazonIdentityManagementClient.deleteUser` permintaan dengan `UpdateUserRequest` objek yang ditetapkan dengan nama pengguna untuk dihapus.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteConflictException;
import com.amazonaws.services.identitymanagement.model.DeleteUserRequest;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DeleteUserRequest request = new DeleteUserRequest()
    .withUserName(username);

try {
    iam.deleteUser(request);
} catch (DeleteConflictException e) {
    System.out.println("Unable to delete user. Verify user is not" +
        " associated with any resources");
    throw e;
}
```

Lihat [contoh lengkapnya](#) GitHub.

Informasi Selengkapnya

- [Pengguna IAM](#) dalam Panduan IAM Pengguna
- [Mengelola Pengguna IAM](#) di Panduan IAM Pengguna
- [CreateUser](#) dalam Referensi IAM API
- [ListUsers](#) dalam Referensi IAM API
- [UpdateUser](#) dalam Referensi IAM API

- [DeleteUser](#) dalam Referensi IAM API

Menggunakan Alias Akun IAM

Jika Anda ingin URL tersebut agar halaman masuk Anda berisi nama perusahaan Anda atau pengenal ramahAkun AWS lainnya, Anda dapat membuat alias akunAkun AWS.

Note

AWS mendukung tepat satu alias akun per akun.

Membuat Alias akun

Untuk membuat alias akun, panggil `createAccountAlias` metode dengan [CreateAccountAliasRequest](#) objek yang berisi nama alias. `AmazonIdentityManagementClient`

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateAccountAliasRequest;
import com.amazonaws.services.identitymanagement.model.CreateAccountAliasResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

CreateAccountAliasRequest request = new CreateAccountAliasRequest()
    .withAccountAlias(alias);

CreateAccountAliasResult response = iam.createAccountAlias(request);
```

Lihat [contoh lengkapnya](#) GitHub.

Alias akun

Untuk mencantumkan alias akun Anda, jika ada, panggil `listAccountAliases` metode tersebut `AmazonIdentityManagementClient`.

Note

Yang dikembalikan [ListAccountAliasesResult](#) mendukung metode yang sama `getIsTruncated` dan seperti `getMarker` metode AWS SDK for Java daftar lainnya, tetapi hanya Akun AWS dapat memiliki satu alias akun.

impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListAccountAliasesResult;
```

kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

ListAccountAliasesResult response = iam.listAccountAliases();

for (String alias : response.getAccountAliases()) {
    System.out.printf("Retrieved account alias %s", alias);
}
```

lihat [contoh lengkapnya](#) GitHub.

Menghapus.

Untuk menghapus alias akun Anda, panggil `deleteAccountAlias` metode tersebut `AmazonIdentityManagementClient`. Saat menghapus alias akun, Anda harus memberikan namanya menggunakan [DeleteAccountAliasRequest](#) objek.

impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteAccountAliasRequest;
import com.amazonaws.services.identitymanagement.model.DeleteAccountAliasResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DeleteAccountAliasRequest request = new DeleteAccountAliasRequest()
    .withAccountAlias(alias);

DeleteAccountAliasResult response = iam.deleteAccountAlias(request);
```

Lihat [contoh lengkapnya](#) GitHub.

Informasi Selengkapnya

- [IDAWS Akun Anda dan Aliasnya](#) di Panduan IAM Pengguna
- [CreateAccountAlias](#) dalam Referensi IAM API
- [ListAccountAliases](#) dalam Referensi IAM API
- [DeleteAccountAlias](#) dalam Referensi IAM API

Cara menggunakan kebijakan IAM

Membuat Kebijakan

Untuk membuat kebijakan baru, berikan nama kebijakan dan dokumen kebijakan yang diformat JSON dalam [CreatePolicyRequest](#) ke `AmazonIdentityManagementClient.createPolicy` metode.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreatePolicyRequest;
import com.amazonaws.services.identitymanagement.model.CreatePolicyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

CreatePolicyRequest request = new CreatePolicyRequest()
    .withPolicyName(policy_name)
    .withPolicyDocument(POLICY_DOCUMENT);
```

```
CreatePolicyResult response = iam.createPolicy(request);
```

Dokumen kebijakan IAM adalah string JSON dengan [sintaks yang terdokumentasi dengan baik](#). Berikut ini adalah contoh yang menyediakan akses untuk membuat permintaan tertentu DynamoDB.

```
public static final String POLICY_DOCUMENT =
    "{" +
    "  \"Version\": \"2012-10-17\", " +
    "  \"Statement\": [ " +
    "    { " +
    "      \"Effect\": \"Allow\", " +
    "      \"Action\": \"logs:CreateLogGroup\", " +
    "      \"Resource\": \"%s\" " +
    "    }, " +
    "    { " +
    "      \"Effect\": \"Allow\", " +
    "      \"Action\": [ " +
    "        \"dynamodb:DeleteItem\", " +
    "        \"dynamodb:GetItem\", " +
    "        \"dynamodb:PutItem\", " +
    "        \"dynamodb:Scan\", " +
    "        \"dynamodb:UpdateItem\" " +
    "      ], " +
    "      \"Resource\": \"RESOURCE_ARN\" " +
    "    } " +
    "  ] " +
    "}";
```

Lihat [Lengkapi Contoh](#) di GitHub.

Mendapatkan kebijakan

Untuk mengambil kebijakan yang ada, hubungi `AmazonIdentityManagementClient`'s `getPolicy` metode, menyediakan ARN kebijakan dalam [GetPolicyRequest](#) objek.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetPolicyRequest;
```



```
import com.amazonaws.services.identitymanagement.model.GetPolicyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

GetPolicyRequest request = new GetPolicyRequest()
    .withPolicyArn(policy_arn);

GetPolicyResult response = iam.getPolicy(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Melampirkan Kebijakan Peran

Anda dapat melampirkan kebijakan ke IAM http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html[role] dengan memanggil

`AmazonIdentityManagementClient.attachRolePolicy` metode, menyediakannya dengan nama peran dan kebijakan ARN dalam [AttachRolePolicyRequest](#).

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.AttachRolePolicyRequest;
import com.amazonaws.services.identitymanagement.model.AttachedPolicy;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

AttachRolePolicyRequest attach_request =
    new AttachRolePolicyRequest()
        .withRoleName(role_name)
        .withPolicyArn(POLICY_ARN);

iam.attachRolePolicy(attach_request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Daftar Kebijakan Peran Terlampir

Buat daftar kebijakan terlampir pada peran dengan memanggil `AmazonIdentityManagementClient`'s `listAttachedRolePolicies` metode.

Dibutuhkan [ListAttachedRolePoliciesRequest](#) objek yang berisi nama peran untuk daftar kebijakan untuk.

Panggil `getAttachedPolicies` pada dikembalikan [ListAttachedRolePoliciesResult](#) keberatan untuk mendapatkan daftar kebijakan terlampir. Hasil dapat dipotong; jika `ListAttachedRolePoliciesResult` objek `getIsTruncated` pengembalian `true`, PANGGIL `ListAttachedRolePoliciesRequest` objek `setMarker` metode dan menggunakannya untuk memanggil `listAttachedRolePolicies` lagi untuk mendapatkan batch berikutnya hasil.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListAttachedRolePoliciesRequest;
import com.amazonaws.services.identitymanagement.model.ListAttachedRolePoliciesResult;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

ListAttachedRolePoliciesRequest request =
    new ListAttachedRolePoliciesRequest()
        .withRoleName(role_name);

List<AttachedPolicy> matching_policies = new ArrayList<>();

boolean done = false;

while(!done) {
    ListAttachedRolePoliciesResult response =
        iam.listAttachedRolePolicies(request);

    matching_policies.addAll(
        response.getAttachedPolicies()
```

```
        .stream()
        .filter(p -> p.getPolicyName().equals(role_name))
        .collect(Collectors.toList());

    if(!response.getIsTruncated()) {
        done = true;
    }
    request.setMarker(response.getMarker());
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Melepaskan kebijakan peran

Untuk melepaskan kebijakan dari peran, panggil `AmazonIdentityManagementClient`'s `detachRolePolicy` metode, menyediakannya dengan nama peran dan kebijakan ARN dalam [DetachRolePolicyRequest](#).

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DetachRolePolicyRequest;
import com.amazonaws.services.identitymanagement.model.DetachRolePolicyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DetachRolePolicyRequest request = new DetachRolePolicyRequest()
    .withRoleName(role_name)
    .withPolicyArn(policy_arn);

DetachRolePolicyResult response = iam.detachRolePolicy(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Informasi Selengkapnya

- [Gambaran umum dari kebijakan IAM](#) di IAM Panduan Pengguna.

- [AWSReferensi kebijakan IAM](#) di [IAMPanduan Pengguna](#).
- [CreatePolicy](#) dalam Referensi API IAM
- [GetPolicy](#) dalam Referensi API IAM
- [AttachRolePolicy](#) dalam Referensi API IAM
- [ListAttachedRolePolicies](#) dalam Referensi API IAM
- [DetachRolePolicy](#) dalam Referensi API IAM

Cara menggunakan Sertifikat Server IAM

Untuk mengaktifkan koneksi HTTPS ke situs web atau aplikasi Anda AWS, Anda membutuhkan SSL/TLS sertifikat server. Anda dapat menggunakan sertifikat server yang disediakan oleh AWS Certificate Manager atau salah satu yang Anda peroleh dari penyedia eksternal.

Kami menyarankan agar Anda menggunakan ACM untuk menyediakan, mengelola, dan menerapkan sertifikat server Anda. Dengan ACM Anda dapat meminta sertifikat, men-deploy nya ke Anda AWS sumber daya, dan membiarkan ACM menangani pembaruan sertifikat untuk Anda. Sertifikat yang disediakan oleh ACM gratis. Untuk informasi selengkapnya tentang ACM, lihat [Panduan Pengguna ACM](#).

Mendapatkan Sertifikat Server

Anda dapat mengambil sertifikat server dengan memanggil `AmazonIdentityManagementClient's getServerCertificate` metode, melewatinya [GetServerCertificateRequest](#) dengan nama sertifikat.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetServerCertificateRequest;
import com.amazonaws.services.identitymanagement.model.GetServerCertificateResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

GetServerCertificateRequest request = new GetServerCertificateRequest()
```

```
.withServerCertificateName(cert_name);

GetServerCertificateResult response = iam.getServerCertificate(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Sertifikat Server

Untuk mencantumkan sertifikat server Anda, hubungi `AmazonIdentityManagementClient`'s `listServerCertificates` metode dengan [ListServerCertificatesRequest](#). Ini menghasilkan [ListServerCertificatesResult](#).

Panggil yang dikembalikan `ListServerCertificateResult` objek `getServerCertificateMetadataList` metode untuk mendapatkan daftar [ServerCertificateMetadata](#) objek yang dapat Anda gunakan untuk mendapatkan informasi tentang setiap sertifikat.

Hasil dapat dipotong; jika `ListServerCertificateResult` objek `getIsTruncated` pengembalian metode `true`, hubungi `ListServerCertificatesRequest` objek `setMarker` metode dan menggunakannya untuk memanggil `listServerCertificates` lagi untuk mendapatkan batch berikutnya hasil.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListServerCertificatesRequest;
import com.amazonaws.services.identitymanagement.model.ListServerCertificatesResult;
import com.amazonaws.services.identitymanagement.model.ServerCertificateMetadata;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

boolean done = false;
ListServerCertificatesRequest request =
    new ListServerCertificatesRequest();

while(!done) {
```

```
ListServerCertificatesResult response =
    iam.listServerCertificates(request);

for(ServerCertificateMetadata metadata :
    response.getServerCertificateMetadataList()) {
    System.out.printf("Retrieved server certificate %s",
        metadata.getServerCertificateName());
}

request.setMarker(response.getMarker());

if(!response.getIsTruncated()) {
    done = true;
}
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Memperbarui Sertifikat Server

Anda dapat memperbarui nama atau jalur sertifikat server dengan memanggil `AmazonIdentityManagementClient.updateServerCertificate` metode.

Dibutuhkan [UpdateServerCertificateRequest](#) objek diatur dengan nama server sertifikat saat ini dan baik nama baru atau jalur baru untuk digunakan.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.UpdateServerCertificateRequest;
import com.amazonaws.services.identitymanagement.model.UpdateServerCertificateResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

UpdateServerCertificateRequest request =
    new UpdateServerCertificateRequest()
        .withServerCertificateName(cur_name)
        .withNewServerCertificateName(new_name);
```

```
UpdateServerCertificateResult response =  
    iam.updateServerCertificate(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Menghapus Sertifikat Server

Untuk menghapus sertifikat server, hubungi `AmazonIdentityManagementClient's deleteServerCertificate` metode dengan [DeleteServerCertificateRequest](#) berisi nama sertifikat.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;  
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;  
import com.amazonaws.services.identitymanagement.model.DeleteServerCertificateRequest;  
import com.amazonaws.services.identitymanagement.model.DeleteServerCertificateResult;
```

Kode

```
final AmazonIdentityManagement iam =  
    AmazonIdentityManagementClientBuilder.defaultClient();  
  
DeleteServerCertificateRequest request =  
    new DeleteServerCertificateRequest()  
        .withServerCertificateName(cert_name);  
  
DeleteServerCertificateResult response =  
    iam.deleteServerCertificate(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Informasi Selengkapnya

- [Bekerja dengan Sertifikat Server](#) di IAM Panduan Pengguna
- [GetServerCertificate](#) dalam Referensi API IAM
- [ListServerCertificates](#) dalam Referensi API IAM
- [UpdateServerCertificate](#) dalam Referensi API IAM
- [DeleteServerCertificate](#) dalam Referensi API IAM
- [Panduan Pengguna ACM](#)

LambdaContoh MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrogramanLambdamenggunakanAWS SDK for Java.

Note

Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parameter[kode contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk membangun dan menjalankan.

Topik

- [Memanggil, Listing, dan MenghapusLambdaFungsi](#)

Memanggil, Listing, dan MenghapusLambdaFungsi

Bagian ini menyediakan contoh pemrograman denganLambdalayanan klien dengan menggunakanAWS SDK for Java. Untuk mempelajari cara membuatLambdafungsi, lihat[Cara MembuatAWS Lambdafungsi](#).

Topik

- [Mengaktifkan fungsi](#)
- [Fungsi daftar](#)
- [Menghapus fungsi](#)

Mengaktifkan fungsi

Anda dapat memanggilLambdaberfungsi dengan membuat[AWSLambda](#)objek dan memohoninvoke metode. Buat[InvokeRequest](#)objek untuk menentukan informasi tambahan seperti nama fungsi dan payload untuk lulus keLambdafungsi. Nama fungsi muncul sebagaiarn:aws:lambda:us-east-1:5556330391:fungsi: helloFunction. Anda dapat mengambil nilai dengan melihat fungsi diAWS Management Console.

Untuk melewatkan data muatan ke fungsi, panggil[InvokeRequest](#)objekwithPayloadMetode dan menentukan String dalam format JSON, seperti yang ditunjukkan pada contoh kode berikut.

IMPOR

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.lambda.AWSLambda;
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;
import com.amazonaws.services.lambda.model.InvokeRequest;
import com.amazonaws.services.lambda.model.InvokeResult;
import com.amazonaws.services.lambda.model.ServiceException;

import java.nio.charset.StandardCharsets;
```

Kode

Contoh kode berikut menunjukkan cara memanggil Lambda fungsi.

```
String functionName = args[0];

InvokeRequest invokeRequest = new InvokeRequest()
    .withFunctionName(functionName)
    .withPayload("{\n" +
        "  \"Hello \": \"Paris\",\n" +
        "  \"countryCode\": \"FR\"\n" +
        "}");
InvokeResult invokeResult = null;

try {
    AWSLambda awsLambda = AWSLambdaClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(Regions.US_WEST_2).build();

    invokeResult = awsLambda.invoke(invokeRequest);

    String ans = new String(invokeResult.getPayload().array(),
        StandardCharsets.UTF_8);

    //write out the return value
    System.out.println(ans);

} catch (ServiceException e) {
    System.out.println(e);
}
```

```
System.out.println(invokeResult.getStatusCode());
```

Lihat contoh lengkap [GitHub](#).

Fungsi daftar

Membangun [AWSLambda](#) objek dan memohon `listFunctions` metode. Metode ini mengembalikan [ListFunctionsResult](#) objek. Anda dapat memanggil objek ini `getFunctions` metode untuk mengembalikan daftar [FunctionConfiguration](#) objek. Anda dapat iterate melalui daftar untuk mengambil informasi tentang fungsi. Sebagai contoh, contoh kode Java berikut menunjukkan cara mendapatkan nama fungsi masing-masing.

IMPOR

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.lambda.AWSLambda;
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;
import com.amazonaws.services.lambda.model.FunctionConfiguration;
import com.amazonaws.services.lambda.model.ListFunctionsResult;
import com.amazonaws.services.lambda.model.ServiceException;
import java.util.Iterator;
import java.util.List;
```

Kode

Contoh kode Java berikut mendemonstrasikan cara mengambil daftar `Lambda` nama fungsi.

```
ListFunctionsResult functionResult = null;

try {
    AWSLambda awsLambda = AWSLambdaClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(Regions.US_WEST_2).build();

    functionResult = awsLambda.listFunctions();

    List<FunctionConfiguration> list = functionResult.getFunctions();

    for (Iterator iter = list.iterator(); iter.hasNext(); ) {
        FunctionConfiguration config = (FunctionConfiguration)iter.next();
```

```
        System.out.println("The function name is "+config.getFunctionName());
    }

    } catch (ServiceException e) {
        System.out.println(e);
    }
}
```

Lihat contoh lengkap [GitHub](#).

Menghapus fungsi

Membangun [AWSLambda](#) objek dan memohon `deleteFunction` metode.

Buat [DeleteFunctionRequest](#) objek dan menyebarkannya ke `deleteFunction` metode. Objek ini berisi informasi seperti nama fungsi untuk menghapus. Nama fungsi muncul sebagai `arn:aws:lambda:us-east-1:5556330391:function:helloFunction`. Anda dapat mengambil nilai dengan melihat fungsi di AWS Management Console.

IMPOR

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.lambda.AWSLambda;
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;
import com.amazonaws.services.lambda.model.ServiceException;
import com.amazonaws.services.lambda.model.DeleteFunctionRequest;
```

Kode

Kode Java berikut menunjukkan cara menghapus `Lambda` fungsi.

```
String functionName = args[0];
try {
    AWSLambda awsLambda = AWSLambdaClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(Regions.US_WEST_2).build();

    DeleteFunctionRequest delFunc = new DeleteFunctionRequest();
    delFunc.withFunctionName(functionName);

    //Delete the function
    awsLambda.deleteFunction(delFunc);
    System.out.println("The function is deleted");
}
```

```
    } catch (ServiceException e) {  
        System.out.println(e);  
    }  
}
```

Lihat contoh lengkap [GitHub](#).

Amazon Pinpoint Contoh Menggunakan AWS SDK for Java

Bagian ini menyediakan contoh pemrograman [Amazon Pinpoint](#) menggunakan [AWS SDK for Java](#).

Note

Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parameter [contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk membangun dan menjalankan.

Topik

- [Membuat dan Menghapus Aplikasi Amazon Pinpoint](#)
- [Membuat Endpoint di Amazon Pinpoint](#)
- [Membuat Segmen di Amazon Pinpoint](#)
- [Membuat Kampanye di Amazon Pinpoint](#)
- [Saluran di Amazon Pinpoint](#)

Membuat dan Menghapus Aplikasi Amazon Pinpoint

Aplikasi adalah Amazon Pinpoint proyek di mana Anda mendefinisikan penonton untuk aplikasi yang berbeda, dan Anda melibatkan audiens ini dengan pesan disesuaikan. Contoh di halaman ini menunjukkan cara membuat aplikasi baru atau menghapus aplikasi yang sudah ada.

Buat Aplikasi

Membuat aplikasi baru di Amazon Pinpoint dengan memberikan nama aplikasi ke [CreateAppRequest](#) objek, dan kemudian melewati objek itu ke `AmazonPinpointClient.createApp` metode.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateAppRequest;
import com.amazonaws.services.pinpoint.model.CreateAppResult;
import com.amazonaws.services.pinpoint.model.CreateApplicationRequest;
```

Kode

```
CreateApplicationRequest appRequest = new CreateApplicationRequest()
    .withName(appName);

CreateAppRequest request = new CreateAppRequest();
request.withCreateApplicationRequest(appRequest);
CreateAppResult result = pinpoint.createApp(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Menghapus Aplikasi

Untuk menghapus aplikasi, hubungi `AmazonPinpointClient.deleteAppRequest` dengan [DeleteAppRequest](#) objek yang diatur dengan nama aplikasi untuk dihapus.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
```

Kode

```
DeleteAppRequest deleteRequest = new DeleteAppRequest()
    .withApplicationId(appID);

pinpoint.deleteApp(deleteRequest);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Informasi Selengkapnya

- [Aplikasi](#) di dalam `Amazon Pinpoint Referensi API`
- [Aplikasi](#) di dalam `Amazon Pinpoint Referensi API`

Membuat Endpoint diAmazon Pinpoint

Sebuah titik akhir secara unik mengidentifikasi perangkat pengguna tempat Anda dapat mengirim notifikasi pushAmazon Pinpoint. Jika aplikasi Anda diaktifkanAmazon Pinpointdukungan, aplikasi Anda secara otomatis mendaftarkan titik akhir denganAmazon Pinpointsaat pengguna baru membuka aplikasi Anda. Contoh berikut menunjukkan cara menambahkan endpoint baru secara pemrograman.

Membuat Endpoint

Buat endpoint baru diAmazon Pinpointdengan menyediakan data endpoint dalam [EndpointRequest](#) objek.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.UpdateEndpointRequest;
import com.amazonaws.services.pinpoint.model.UpdateEndpointResult;
import com.amazonaws.services.pinpoint.model.EndpointDemographic;
import com.amazonaws.services.pinpoint.model.EndpointLocation;
import com.amazonaws.services.pinpoint.model.EndpointRequest;
import com.amazonaws.services.pinpoint.model.EndpointResponse;
import com.amazonaws.services.pinpoint.model.EndpointUser;
import com.amazonaws.services.pinpoint.model.GetEndpointRequest;
import com.amazonaws.services.pinpoint.model.GetEndpointResult;
```

Kode

```
HashMap<String, List<String>> customAttributes = new HashMap<>();
List<String> favoriteTeams = new ArrayList<>();
favoriteTeams.add("Lakers");
favoriteTeams.add("Warriors");
customAttributes.put("team", favoriteTeams);

EndpointDemographic demographic = new EndpointDemographic()
    .withAppVersion("1.0")
    .withMake("apple")
    .withModel("iPhone")
    .withModelVersion("7")
    .withPlatform("ios")
```

```
.withPlatformVersion("10.1.1")
.withTimezone("America/Los_Angeles");

EndpointLocation location = new EndpointLocation()
    .withCity("Los Angeles")
    .withCountry("US")
    .withLatitude(34.0)
    .withLongitude(-118.2)
    .withPostalCode("90068")
    .withRegion("CA");

Map<String,Double> metrics = new HashMap<>();
metrics.put("health", 100.00);
metrics.put("luck", 75.00);

EndpointUser user = new EndpointUser()
    .withUserId(UUID.randomUUID().toString());

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());

EndpointRequest endpointRequest = new EndpointRequest()
    .withAddress(UUID.randomUUID().toString())
    .withAttributes(customAttributes)
    .withChannelType("APNS")
    .withDemographic(demographic)
    .withEffectiveDate(nowAsISO)
    .withLocation(location)
    .withMetrics(metrics)
    .withOptOut("NONE")
    .withRequestId(UUID.randomUUID().toString())
    .withUser(user);
```

Kemudian membuat [UpdateEndpointRequest](#) objek dengan itu `EndpointRequest` objek. Akhirnya, lulus `UpdateEndpointRequest` keberatan dengan `AmazonPinpointClient.updateEndpoint` metode.

Kode

```
UpdateEndpointRequest updateEndpointRequest = new UpdateEndpointRequest()
    .withApplicationId(appId)
    .withEndpointId(endpointId)
    .withEndpointRequest(endpointRequest);
```

```
UpdateEndpointResult updateEndpointResponse =
    client.updateEndpoint(updateEndpointRequest);
System.out.println("Update Endpoint Response: " +
    updateEndpointResponse.getMessageBody());
```

Lihat [contoh lengkap](#) di GitHub.

Informasi Selengkapnya

- [Menambahkan Endpoint](#) di Amazon Pinpoint Panduan Pengembang
- [Titik akhir](#) di Amazon Pinpoint Referensi API

Membuat Segmen di Amazon Pinpoint

Segmen pengguna mewakili subset pengguna Anda yang didasarkan pada karakteristik bersama, seperti bagaimana baru-baru ini pengguna membuka aplikasi atau perangkat mana yang mereka gunakan. Contoh berikut menunjukkan cara menentukan segmen pengguna.

Buat Segmen

Buat segmen baru di Amazon Pinpoint dengan mendefinisikan dimensi segmen dalam [SegmentDimensions](#) objek.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateSegmentRequest;
import com.amazonaws.services.pinpoint.model.CreateSegmentResult;
import com.amazonaws.services.pinpoint.model.AttributeDimension;
import com.amazonaws.services.pinpoint.model.AttributeType;
import com.amazonaws.services.pinpoint.model.RecencyDimension;
import com.amazonaws.services.pinpoint.model.SegmentBehaviors;
import com.amazonaws.services.pinpoint.model.SegmentDemographics;
import com.amazonaws.services.pinpoint.model.SegmentDimensions;
import com.amazonaws.services.pinpoint.model.SegmentLocation;
import com.amazonaws.services.pinpoint.model.SegmentResponse;
import com.amazonaws.services.pinpoint.model.WriteSegmentRequest;
```

Kode


```

Pinpoint pinpoint =
    AmazonPinpointClientBuilder.standard().withRegion(Regions.US_EAST_1).build();
Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
segmentAttributes.put("Team", new
    AttributeDimension().withAttributeType(AttributeType.INCLUSIVE).withValues("Lakers"));

SegmentBehaviors segmentBehaviors = new SegmentBehaviors();
SegmentDemographics segmentDemographics = new SegmentDemographics();
SegmentLocation segmentLocation = new SegmentLocation();

RecencyDimension recencyDimension = new RecencyDimension();
recencyDimension.withDuration("DAY_30").withRecencyType("ACTIVE");
segmentBehaviors.setRecency(recencyDimension);

SegmentDimensions dimensions = new SegmentDimensions()
    .withAttributes(segmentAttributes)
    .withBehavior(segmentBehaviors)
    .withDemographic(segmentDemographics)
    .withLocation(segmentLocation);

```

Berikutnya mengatur [SegmentDimensions](#) objek dalam [WriteSegmentRequest](#), yang pada gilirannya digunakan untuk membuat [CreateSegmentRequest](#) objek. Kemudian lewati `CreateSegmentRequest` keberatan dengan `AmazonPinpointClient.createSegment` metode.

Kode

```

WriteSegmentRequest writeSegmentRequest = new WriteSegmentRequest()
    .withName("MySegment").withDimensions(dimensions);

CreateSegmentRequest createSegmentRequest = new CreateSegmentRequest()
    .withApplicationId(appId).withWriteSegmentRequest(writeSegmentRequest);

CreateSegmentResult createSegmentResult = client.createSegment(createSegmentRequest);

```

Lihat [contoh lengkap](#) di GitHub.

Informasi Selengkapnya

- [Amazon Pinpoint Segmen](#) di Amazon Pinpoint Panduan Pengguna
- [Membuat Segmen](#) di Amazon Pinpoint Panduan Pengembang
- [Segmen](#) di Amazon Pinpoint Referensi API

- [Segment](#) di Amazon Pinpoint Referensi API

Membuat Kampanye di Amazon Pinpoint

Anda dapat menggunakan kampanye untuk membantu meningkatkan keterlibatan antara aplikasi dan pengguna Anda. Anda dapat membuat kampanye untuk menjangkau segmen pengguna tertentu dengan pesan khusus atau promosi khusus. Contoh ini menunjukkan cara membuat kampanye standar baru yang mengirimkan pemberitahuan push kustom ke segmen tertentu.

Membuat Kampanye

Sebelum membuat kampanye baru, Anda harus menentukan [Jadwal](#) dan [Pesan](#) dan menetapkan nilai-nilai ini dalam [WriteCampaignRequest](#) objek.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateCampaignRequest;
import com.amazonaws.services.pinpoint.model.CreateCampaignResult;
import com.amazonaws.services.pinpoint.model.Action;
import com.amazonaws.services.pinpoint.model.CampaignResponse;
import com.amazonaws.services.pinpoint.model.Message;
import com.amazonaws.services.pinpoint.model.MessageConfiguration;
import com.amazonaws.services.pinpoint.model.Schedule;
import com.amazonaws.services.pinpoint.model.WriteCampaignRequest;
```

Kode

```
Schedule schedule = new Schedule()
    .withStartTime("IMMEDIATE");

Message defaultMessage = new Message()
    .withAction(Action.OPEN_APP)
    .withBody("My message body.")
    .withTitle("My message title.");

MessageConfiguration messageConfiguration = new MessageConfiguration()
    .withDefaultMessage(defaultMessage);

WriteCampaignRequest request = new WriteCampaignRequest()
```

```
.withDescription("My description.")
.withSchedule(schedule)
.withSegmentId(segmentId)
.withName("MyCampaign")
.withMessageConfiguration(messageConfiguration);
```

Kemudian buat kampanye baru Amazon Pinpoint [WriteCampaignRequest](#) dengan menyediakan konfigurasi kampanye ke [CreateCampaignRequest](#) objek. Akhirnya, lewati `CreateCampaignRequest` objek ke `AmazonPinpointClient.createCampaign` metode.

Kode

```
CreateCampaignRequest createCampaignRequest = new CreateCampaignRequest()
    .withApplicationId(appId).withWriteCampaignRequest(request);

CreateCampaignResult result = client.createCampaign(createCampaignRequest);
```

Lihat [contoh lengkapnya](#) GitHub.

Informasi Selengkapnya

- [Amazon Pinpoint Kampanye](#) di Panduan Amazon Pinpoint Pengguna
- [Membuat Kampanye](#) di Panduan Amazon Pinpoint Pengembang
- [Kampanye](#) di Referensi Amazon Pinpoint API
- [Kampanye](#) di Referensi Amazon Pinpoint API
- [Aktivitas Kampanye](#) di Referensi Amazon Pinpoint API
- [Versi Kampanye](#) di Referensi Amazon Pinpoint API
- [Versi Kampanye](#) di Referensi Amazon Pinpoint API

Saluran di Amazon Pinpoint

Saluran mendefinisikan jenis platform tempat Anda dapat mengirimkan pesan. Contoh ini menunjukkan cara menggunakan saluran APN untuk mengirim pesan.

Perbarui Saluran

Aktifkan saluran di Amazon Pinpoint dengan menyediakan ID aplikasi dan objek permintaan dari jenis saluran yang ingin Anda perbarui. Contoh ini memperbarui saluran APN, yang

memerlukan [APNSChannelRequest](#) objek. Set ini di [UpdateApnsChannelRequest](#) dan berikan objek itu ke `AmazonPinpointClient.updateApnsChannel` metode.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.APNSChannelRequest;
import com.amazonaws.services.pinpoint.model.APNSChannelResponse;
import com.amazonaws.services.pinpoint.model.GetApnsChannelRequest;
import com.amazonaws.services.pinpoint.model.GetApnsChannelResult;
import com.amazonaws.services.pinpoint.model.UpdateApnsChannelRequest;
import com.amazonaws.services.pinpoint.model.UpdateApnsChannelResult;
```

Kode

```
APNSChannelRequest request = new APNSChannelRequest()
    .withEnabled(enabled);

UpdateApnsChannelRequest updateRequest = new UpdateApnsChannelRequest()
    .withAPNSChannelRequest(request)
    .withApplicationId(appId);
UpdateApnsChannelResult result = client.updateApnsChannel(updateRequest);
```

Lihat [contoh lengkap](#) di GitHub.

Informasi Selengkapnya

- [Amazon PinpointChannel](#) di dalam [Amazon Pinpoint Panduan Pengguna](#)
- [Saluran ADM](#) di dalam [Amazon Pinpoint Referensi API](#)
- [Saluran](#) di dalam [Amazon Pinpoint Referensi API](#)
- [Saluran APN](#) di dalam [Amazon Pinpoint Referensi API](#)
- [Saluran APN](#) di dalam [Amazon Pinpoint Referensi API](#)
- [Saluran Sandbox VoIP APN](#) di dalam [Amazon Pinpoint Referensi API](#)
- [Saluran](#) di dalam [Amazon Pinpoint Referensi API](#)
- [Saluran](#) di dalam [Amazon Pinpoint Referensi API](#)
- [Saluran GCM](#) di dalam [Amazon Pinpoint Referensi API](#)
- [Saluran SMS](#) di dalam [Amazon Pinpoint Referensi API](#)

Amazon S3 Contoh Menggunakan AWS SDK for Java

Bagian ini menyediakan contoh pemrograman [Amazon S3](#) menggunakan [AWS SDK for Java](#).

Note

Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parameter [contoh kode lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk membangun dan menjalankan.

Topik

- [Membuat, Daftar, dan Menghapus Amazon S3 Bucket](#)
- [Melakukan Operasi pada Amazon S3 Objek](#)
- [Mengelola Amazon S3 Akses Izin untuk Bucket dan Objek](#)
- [Mengelola akses ke Amazon S3 Bucket Menggunakan Kebijakan ember](#)
- [Menggunakan TransferManager untuk Amazon S3 Operasi](#)
- [Mengonfigurasi Amazon S3 Bucket sebagai Situs Web](#)
- [Gunakan Amazon S3 Enkripsi sisi klien](#)

Membuat, Daftar, dan Menghapus Amazon S3 Bucket

Setiap objek (file) di Amazon S3 harus berada dalam ember, yang merupakan koleksi (wadah) objek. Setiap ember dikenal oleh kunci (nama), yang harus unik. Untuk informasi mendetail tentang bucket dan konfigurasinya, lihat [Bekerja dengan Amazon S3 Bucket](#) di dalam Amazon Simple Storage Service Panduan Pengguna.

Note

Praktik terbaik

Kami sarankan Anda mengaktifkan [AbortIncompleteMultipartUpload](#) aturan siklus hidup di Amazon S3 ember.

Aturan ini mengarahkan Amazon S3 untuk membatalkan unggahan multi-bagian yang tidak selesai dalam jumlah hari tertentu setelah dimulai. Ketika batas waktu yang ditetapkan

terlampai, Amazon S3 membatalkan upload dan kemudian menghapus data upload yang tidak lengkap.

Untuk informasi selengkapnya, lihat [Konfigurasi Siklus Hidup untuk Bucket dengan Versioning](#) di dalam [Amazon S3 Panduan Pengguna](#).

Note

Contoh kode ini berasumsi bahwa Anda memahami materi di [Menggunakan AWS SDK for Java](#) dan telah dikonfigurasi default AWS kredensi menggunakan informasi di [Mengatur AWS Kredensial dan Wilayah untuk Pembangunan](#).

Buat Bucket

Gunakan klien Amazon S3 `createBucket` metode. Yang baru `Bucket` dikembalikan.

Parameter `createBucket` metode akan meningkatkan pengecualian jika ember sudah ada.

Note

Untuk memeriksa apakah bucket sudah ada sebelum mencoba membuat satu dengan nama yang sama, panggil `doesBucketExist` metode. Ini akan kembali `true` jika ember ada, dan `false` sebaliknya.

Impor

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;
```

Kode

```
if (s3.doesBucketExistV2(bucket_name)) {
```

```
        System.out.format("Bucket %s already exists.\n", bucket_name);
        b = getBucket(bucket_name);
    } else {
        try {
            b = s3.createBucket(bucket_name);
        } catch (AmazonS3Exception e) {
            System.err.println(e.getErrorMessage());
        }
    }
}
return b;
```

Lihat [Lengkapi Contoh](#) di GitHub.

Buat Daftar Bucket

Gunakan klien AmazonS3 `listBucket` metode. Jika berhasil, daftar [Bucket](#) dikembalikan.

Impor

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;
```

Kode

```
List<Bucket> buckets = s3.listBuckets();
System.out.println("Your {S3} buckets are:");
for (Bucket b : buckets) {
    System.out.println("* " + b.getName());
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Hapus Bucket

Sebelum Anda dapat menghapus Amazon S3 ember, Anda harus memastikan bahwa ember kosong atau kesalahan akan menghasilkan. Jika Anda memiliki [ember berversi](#), Anda juga harus menghapus objek berversi yang terkait dengan bucket.

Note

Parameter [Lengkapi Contoh](#) mencakup masing-masing langkah ini secara berurutan, memberikan solusi lengkap untuk menghapus Amazon S3 ember dan isinya.

Topik

- [Hapus Objek dari Bucket Unversioned Sebelum Menghapusnya](#)
- [Hapus Objek dari Bucket Berversi Sebelum Menghapusnya](#)
- [Hapus Bucket Kosong](#)

Hapus Objek dari Bucket Unversioned Sebelum Menghapusnya

Gunakan klien AmazonS3 `listObjects` metode untuk mengambil daftar objek dan `deleteObject` untuk menghapus masing-masing.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;
```

Kode

```
System.out.println(" - removing objects from bucket");
ObjectListing object_listing = s3.listObjects(bucket_name);
while (true) {
    for (Iterator<?> iterator =
        object_listing.getObjectSummaries().iterator();
        iterator.hasNext(); ) {
        S3ObjectSummary summary = (S3ObjectSummary) iterator.next();
        s3.deleteObject(bucket_name, summary.getKey());
    }

    // more object_listing to retrieve?
    if (object_listing.isTruncated()) {
```



```
        object_listing = s3.listNextBatchOfObjects(object_listing);
    } else {
        break;
    }
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Hapus Objek dari Bucket Berversi Sebelum Menghapusnya

Jika Anda menggunakan [ember berversi](#), Anda juga perlu menghapus versi tersimpan dari objek dalam ember sebelum ember dapat dihapus.

Menggunakan pola yang mirip dengan yang digunakan saat menghapus objek dalam ember, hapus objek berversi dengan menggunakan klien AmazonS3 `listVersions` metode untuk daftar objek berversi, dan kemudi `deleteVersion` untuk menghapus masing-masing.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;
```

Kode

```
System.out.println(" - removing versions from bucket");
VersionListing version_listing = s3.listVersions(
    new ListVersionsRequest().withBucketName(bucket_name));
while (true) {
    for (Iterator<?> iterator =
        version_listing.getVersionSummaries().iterator();
        iterator.hasNext(); ) {
        S3VersionSummary vs = (S3VersionSummary) iterator.next();
        s3.deleteVersion(
            bucket_name, vs.getKey(), vs.getVersionId());
    }

    if (version_listing.isTruncated()) {
        version_listing = s3.listNextBatchOfVersions(
```

```
        version_listing);  
    } else {  
        break;  
    }  
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Hapus Bucket Kosong

Setelah Anda menghapus objek dari ember (termasuk objek berversi), Anda dapat menghapus bucket itu sendiri dengan menggunakan klien `AmazonS3deleteBucket` metode.

Impor

```
import com.amazonaws.AmazonServiceException;  
import com.amazonaws.regions.Regions;  
import com.amazonaws.services.s3.AmazonS3;  
import com.amazonaws.services.s3.AmazonS3ClientBuilder;  
import com.amazonaws.services.s3.model.*;  
  
import java.util.Iterator;
```

Kode

```
System.out.println(" OK, bucket ready to delete!");  
s3.deleteBucket(bucket_name);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Melakukan Operasi pada Amazon S3 Objek

Sebuah Amazon S3 objek merupakan file atau koleksi data. Setiap objek harus berada dalam [ember](#).

Note

Contoh kode ini mengasumsikan bahwa Anda memahami materi dalam [Menggunakan AWS SDK for Java](#) dan telah mengkonfigurasi AWS kredensial default menggunakan informasi di [Menyiapkan AWS Kredensial dan Wilayah untuk Pengembangan](#).

Topik

- [Meng-unggah Objek](#)
- [Daftar Objek](#)
- [Mengunduh Objek](#)
- [Menyalin, Memindahkan, atau Mengganti Nama Objek](#)
- [Menghapus Objek](#)
- [Hapus Beberapa Objek Sekaligus](#)

Meng-unggah Objek

Gunakan `putObject` metode klien AmazonS3, menyediakan nama bucket, nama kunci, dan file untuk diunggah. Bucket harus ada, atau error akan terjadi.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
```

Kode

```
System.out.format("Uploading %s to S3 bucket %s...\n", file_path, bucket_name);
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.putObject(bucket_name, key_name, new File(file_path));
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) GitHub.

Daftar Objek

Untuk mendapatkan daftar objek dalam bucket, gunakan `listObjects` metode klien AmazonS3, yang menyediakan nama bucket.

`listObjects` Metode mengembalikan sebuah [ObjectListing](#) objek yang menyediakan informasi tentang objek dalam ember. Untuk daftar nama objek (kunci), gunakan `getObjectSummaries`

metode untuk mendapatkan Daftar ObjectSummary objek [S3](#), yang masing-masing mewakili satu objek dalam ember. Kemudian panggil getKey metodenya untuk mengambil nama objek.

Impor

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;
```

Kode

```
System.out.format("Objects in S3 bucket %s:\n", bucket_name);
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
ListObjectsV2Result result = s3.listObjectsV2(bucket_name);
List<S3ObjectSummary> objects = result.getObjectSummaries();
for (S3ObjectSummary os : objects) {
    System.out.println("* " + os.getKey());
}
```

Lihat [contoh lengkapnya](#) GitHub.

Mengunduh Objek

Gunakan getObject metode klien AmazonS3, berikan nama bucket dan objek untuk diunduh. Jika berhasil, metode mengembalikan [S3Object](#). Bucket dan kunci objek yang ditentukan harus ada, atau error akan terjadi.

Anda bisa mendapatkan isi objek dengan memanggil getObjectContentS3Object. Ini mengembalikan [S3 ObjectInputStream](#) yang berperilaku sebagai objek Java InputStream standar.

Contoh berikut mendownload objek dari S3 dan menyimpan isinya ke file (menggunakan nama yang sama dengan kunci objek).

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

```
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import java.io.File;
```

Kode

```
System.out.format("Downloading %s from S3 bucket %s...\n", key_name, bucket_name);
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    S3Object o = s3.getObject(bucket_name, key_name);
    S3ObjectInputStream s3is = o.getObjectContent();
    FileOutputStream fos = new FileOutputStream(new File(key_name));
    byte[] read_buf = new byte[1024];
    int read_len = 0;
    while ((read_len = s3is.read(read_buf)) > 0) {
        fos.write(read_buf, 0, read_len);
    }
    s3is.close();
    fos.close();
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
} catch (FileNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) GitHub.

Menyalin, Memindahkan, atau Mengganti Nama Objek

Anda dapat menyalin sebuah objek dari satu bucket ke bucket lainnya menggunakan `AmazonS3.copyObject`. Dibutuhkan nama bucket untuk menyalin dari, objek untuk menyalin, dan nama bucket tujuan.

Impor

```
import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.regions.Regions;
```

Kode

```
try {
    s3.copyObject(from_bucket, object_key, to_bucket, object_key);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Done!");
```

Lihat [contoh lengkapnya](#) GitHub.

Note

Anda dapat menggunakan `copyObject` dengan [deleteObject](#) untuk memindahkan atau mengganti nama objek, dengan terlebih dahulu menyalin objek ke nama baru (Anda dapat menggunakan bucket yang sama sebagai sumber dan tujuan) dan kemudian menghapus objek dari lokasi lamanya.

Menghapus Objek

Gunakan `deleteObject` metode klien `AmazonS3`, berikan nama bucket dan objek yang akan dihapus. Bucket dan kunci objek yang ditentukan harus ada, atau eror akan terjadi.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Kode

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.deleteObject(bucket_name, object_key);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

```
}
```

Lihat [contoh lengkapnya](#) GitHub.

Hapus Beberapa Objek Sekaligus

Menggunakan `deleteObjects` metode AmazonS3 klien, Anda dapat menghapus beberapa objek dari bucket yang sama dengan meneruskan nama mereka ke link: `sdk-for-java DeleteObjectsRequest /v1/reference/com/amazonaws/services/s3/model/.html` metode.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
```

Kode

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    DeleteObjectsRequest dor = new DeleteObjectsRequest(bucket_name)
        .withKeys(object_keys);
    s3.deleteObjects(dor);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) GitHub.

Mengelola Amazon S3 Akses Izin untuk Bucket dan Objek

Anda dapat menggunakan daftar kontrol akses (ACL) untuk Amazon S3 ember dan objek untuk kontrol halus atas Anda Amazon S3 sumber daya.

Note

Contoh-contoh kode ini berasumsi bahwa Anda memahami materi di [Menggunakan AWS SDK for Java](#) dan telah dikonfigurasi default AWS kredensi menggunakan informasi di [Mengatur AWS Kredensial dan Wilayah untuk Pembangunan](#).

Dapatkan Daftar Kontrol Akses untuk Bucket

Untuk mendapatkan ACL saat ini untuk ember, hubungi `AmazonS3.getBucketAcl` metode, melewatinya nama bucket kueri. Metode ini mengembalikan [AccessControlList](#) objek. Untuk mendapatkan setiap hibah akses dalam daftar, hubungi `getGrantsAsList` metode, yang akan mengembalikan daftar Java standar [Pemberian Izin](#) benda.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.Grant;
```

Kode

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    AccessControlList acl = s3.getBucketAcl(bucket_name);
    List<Grant> grants = acl.getGrantsAsList();
    for (Grant grant : grants) {
        System.out.format(" %s: %s\n", grant.getGrantee().getIdentifier(),
            grant.getPermission().toString());
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Mengatur Access Control List untuk Bucket

Untuk menambah atau memodifikasi izin ke ACL untuk bucket, hubungi `AmazonS3.setBucketAcl` metode. Dibutuhkan [AccessControlList](#) objek yang berisi daftar penerima hibah dan tingkat akses untuk mengatur.

Impor


```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
```

Kode

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    // get the current ACL
    AccessControlList acl = s3.getBucketAcl(bucket_name);
    // set access for the grantee
    EmailAddressGrantee grantee = new EmailAddressGrantee(email);
    Permission permission = Permission.valueOf(access);
    acl.grantPermission(grantee, permission);
    s3.setBucketAcl(bucket_name, acl);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Note

Anda dapat memberikan pengenal unik penerima hibah secara langsung menggunakan [Penerima izin](#) kelas, atau menggunakan [EmailAddressGrantee](#) kelas untuk mengatur penerima hibah melalui email, seperti yang telah kita lakukan di sini.

Lihat [Lengkapi Contoh](#) di GitHub.

Dapatkan Daftar Kontrol Akses untuk Objek

Untuk mendapatkan ACL saat ini untuk sebuah objek, panggil `AmazonS3.getObjectAcl` metode, melewati nama bucket dan nama objek kueri. `LIKE.getBucketAcl`, metode ini mengembalikan [AccessControlList](#) objek yang dapat Anda gunakan untuk memeriksa masing-masing [Pemberian Izin](#).

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.Grant;
```

Kode

```
try {
    AccessControlList acl = s3.getObjectAcl(bucket_name, object_key);
    List<Grant> grants = acl.getGrantsAsList();
    for (Grant grant : grants) {
        System.out.format("  %s: %s\n", grant.getGrantee().getIdentifier(),
            grant.getPermission().toString());
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Mengatur Access Control List untuk Object

Untuk menambah atau memodifikasi izin ke ACL untuk objek, panggil `AmazonS3.setObjectAcl` metode. Dibutuhkan [AccessControlList](#) objek yang berisi daftar penerima hibah dan tingkat akses untuk mengatur.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
```

Kode

```
try {
    // get the current ACL
```

```
AccessControlList acl = s3.getObjectAcl(bucket_name, object_key);
// set access for the grantee
EmailAddressGrantee grantee = new EmailAddressGrantee(email);
Permission permission = Permission.valueOf(access);
acl.grantPermission(grantee, permission);
s3.setObjectAcl(bucket_name, object_key, acl);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
}
```

Note

Anda dapat memberikan pengenal unik penerima hibah secara langsung menggunakan [Penerima izin](#) kelas, atau menggunakan [EmailAddressGrantee](#) kelas untuk mengatur penerima hibah melalui email, seperti yang telah kita lakukan di sini.

Lihat [Lengkapi Contoh](#) di GitHub.

Informasi Selengkapnya

- [acl DAPATKAN ember](#) di Amazon S3 Referensi API
- [acl LETAKKAN ember](#) di Amazon S3 Referensi API
- [acl DAPATKAN Objek](#) di Amazon S3 Referensi API
- [acl LETAKKAN Objek](#) di Amazon S3 Referensi API

Mengelola akses ke Amazon S3 Bucket Menggunakan Kebijakan ember

Anda dapat mengatur, mendapatkan, atau menghapus kebijakan bucket untuk mengelola akses ke Amazon S3 ember.

Mengatur Kebijakan ember

Anda dapat mengatur kebijakan bucket S3 tertentu dengan:

- Memanggil klien `AmazonS3.setBucketPolicy` dan menyediakannya dengan [SetBucketPolicyRequest](#)

- Mengatur kebijakan secara langsung dengan menggunakan `setBucketPolicy` kelebihan beban yang mengambil nama bucket dan teks kebijakan (dalam format JSON)

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
```

Kode

```
s3.setBucketPolicy(bucket_name, policy_text);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Gunakan Kelas Kebijakan untuk Menghasilkan atau Memvalidasi Kebijakan

Saat memberikan kebijakan bucket `setBucketPolicy`, Anda dapat melakukan hal berikut:

- Tentukan kebijakan secara langsung sebagai string teks berformat JSON
- Membangun kebijakan menggunakan [Kebijakan](#) kelas

Dengan menggunakan `Policy` kelas, Anda tidak perlu khawatir tentang benar memformat string teks Anda. Untuk mendapatkan teks kebijakan JSON dari `Policy` kelas, menggunakan `toJson` metode.

Impor

```
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

Kode

```
new Statement(Statement.Effect.Allow)
```

```

        .withPrincipals(Principal.AllUsers)
        .withActions(S3Actions.GetObject)
        .withResources(new Resource(
            "{region-arn}s3:::" + bucket_name + "/*"));
return bucket_policy.toJson();

```

`ParameterPolicy` kelas juga menyediakan `fromJson` metode yang dapat mencoba untuk membangun kebijakan menggunakan passed-in JSON string. Metode memvalidasi untuk memastikan bahwa teks dapat diubah menjadi struktur kebijakan yang valid, dan akan gagal dengan `IllegalArgumentException` jika teks kebijakan tidak valid.

```

Policy bucket_policy = null;
try {
    bucket_policy = Policy.fromJson(file_text.toString());
} catch (IllegalArgumentException e) {
    System.out.format("Invalid policy text in file: \"%s\"",
        policy_file);
    System.out.println(e.getMessage());
}

```

Anda dapat menggunakan teknik ini untuk memvalidasi kebijakan yang Anda baca dari file atau cara lain.

Lihat [Lengkapi Contoh](#) di GitHub.

Dapatkan Kebijakan ember

Untuk mengambil kebijakan untuk Amazon S3 ember, hubungi klien `AmazonS3` `getBucketPolicy` metode, lewat itu nama ember untuk mendapatkan kebijakan dari.

Impor

```

import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

```

Kode

```

try {

```

```
    BucketPolicy bucket_policy = s3.getBucketPolicy(bucket_name);
    policy_text = bucket_policy.getPolicyText();
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Jika bucket bernama tidak ada, jika Anda tidak memiliki akses ke sana, atau jika tidak memiliki kebijakan bucket, sebuah `AmazonServiceException` dilemparkan.

Lihat [Lengkapi Contoh](#) di GitHub.

Menghapus Kebijakan ember

Untuk menghapus kebijakan bucket, hubungi klien AmazonS3 `deleteBucketPolicy`, menyediakannya dengan nama bucket.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
```

Kode

```
try {
    s3.deleteBucketPolicy(bucket_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Metode ini berhasil bahkan jika bucket belum memiliki kebijakan. Jika Anda menentukan nama bucket yang tidak ada atau jika Anda tidak memiliki akses ke bucket, sebuah `AmazonServiceException` dilemparkan.

Lihat [Lengkapi Contoh](#) di GitHub.

Info Selengkapnya

- [Ikhtisar Bahasa Kebijakan](#) di dalam Amazon Simple Storage Service Panduan Pengguna

- [Contoh Kebijakan ember](#) di dalam Amazon Simple Storage Service Panduan Pengguna

Menggunakan TransferManager untuk Amazon S3 Operasi

Anda dapat menggunakan AWS SDK for Java TransferManager kelas untuk andal mentransfer file dari lingkungan lokal ke Amazon S3 dan untuk menyalin objek dari satu lokasi S3 ke lokasi lainnya. TransferManager bisa mendapatkan kemajuan transfer dan jeda atau melanjutkan upload dan download.

Note

Praktik terbaik

Kami sarankan Anda mengaktifkan [AbortIncompleteMultipartUpload](#) aturan siklus hidup Amazon S3 ember.

Aturan ini mengarahkan Amazon S3 untuk membatalkan unggahan multi-bagian yang tidak selesai dalam jumlah hari tertentu setelah dimulai. Ketika batas waktu yang ditetapkan terlampaui, Amazon S3 membatalkan upload dan kemudian menghapus data upload yang tidak lengkap.

Untuk informasi selengkapnya, lihat [Konfigurasi Siklus Hidup untuk Bucket dengan Versioning](#) di Amazon S3 Panduan Pengguna.

Note

Contoh-contoh kode ini berasumsi bahwa Anda memahami materi di [Menggunakan AWS SDK for Java](#) dan telah dikonfigurasi default AWS kredensi menggunakan informasi di [Mengatur AWS Kredensial dan Wilayah untuk Pembangunan](#).

Upload File dan Direktori

TransferManager dapat meng-upload file, daftar file, dan direktori ke setiap Amazon S3 ember yang Anda miliki [sebelumnya dibuat](#).

Topik

- [Meng-unggah File Tunggal](#)
- [Unggah Daftar File](#)

- [Meng-unggah sebuah Direktori](#)

Meng-unggah File Tunggal

Hubungi `TransferManager.upload` metode, menyediakan Amazon S3 nama bucket, kunci (objek) nama, dan Java standar [Berkas](#) objek yang mewakili file untuk meng-unggah.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.MultipleFileUpload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
```

Kode

```
File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Upload xfer = xfer_mgr.upload(bucket_name, key_name, f);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Parameter `upload` metode pengembalian langsung, menyediakan `Upload` objek yang akan digunakan untuk memeriksa status transfer atau menunggu sampai selesai.

Lihat [Tunggu Transfer untuk Menyelesaikan](#) untuk informasi tentang menggunakan `waitForCompletion` untuk berhasil menyelesaikan transfer sebelum memanggil `TransferManager.shutdownNow` metode. Sambil menunggu transfer selesai, Anda dapat memilih atau

mendengarkan pembaruan tentang status dan kemajuannya. Lihat [Dapatkan Status Transfer dan Kemajuan](#) untuk informasi lebih lanjut.

Lihat [Lengkapi Contoh](#) di GitHub.

Unggah Daftar File

Untuk mengunggah beberapa file dalam satu operasi, panggil `TransferManager.uploadFileList`, memberikan hal berikut:

- Sesi Amazon S3 nama bucket
- SEBUAH prefiks kunci untuk menambahkan nama-nama objek yang dibuat (jalan dalam ember di mana untuk menempatkan objek)
- SEBUAH [Berkas](#) objek yang mewakili direktori relatif dari yang untuk membuat path file
- SEBUAH [Daftar](#) objek yang berisi satu set [Berkas](#) objek untuk diunggah

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.MultipleFileUpload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
```

Kode

```
ArrayList<File> files = new ArrayList<File>();
for (String path : file_paths) {
    files.add(new File(path));
}

TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    MultipleFileUpload xfer = xfer_mgr.uploadFileList(bucket_name,
        key_prefix, new File("."), files);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
}
```

```

// or block with Transfer.waitForCompletion()
XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();

```

Lihat [Tunggu Transfer untuk Menyelesaikan](#) untuk informasi tentang menggunakan `waitForCompletion` untuk berhasil menyelesaikan transfer sebelum memanggil `TransferManager.shutdownNow` metode. Sambil menunggu transfer selesai, Anda dapat memilih atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat [Dapatkan Status Transfer dan Kemajuan](#) untuk informasi lebih lanjut.

Parameter [MultipleFileUpload](#) objek yang dikembalikan oleh `uploadFileList` dapat digunakan untuk query negara transfer atau kemajuan. Lihat [Poll Kemajuan Transfer Saat Ini](#) dan [Dapatkan Transfer Progress dengan ProgressListener](#) untuk informasi lebih lanjut.

Anda juga dapat menggunakan `MultipleFileUpload`'s `getSubTransfers` metode untuk mendapatkan `individuUpload` objek untuk setiap file yang ditransfer. Untuk informasi selengkapnya, lihat [Dapatkan Kemajuan Subtransfer](#).

Lihat [Lengkapi Contoh](#) di GitHub.

Meng-unggah sebuah Direktori

Anda dapat menggunakan `TransferManager.uploadDirectory` metode untuk meng-upload seluruh direktori file, dengan pilihan untuk menyalin file dalam subdirektori rekursif. Anda menyediakan Amazon S3 nama ember, key prefix S3, a [Berkas](#) objek yang mewakili direktori lokal untuk menyalin, dan `boolean` nilai yang menunjukkan apakah Anda ingin menyalin subdirektori secara rekursif (benar atau palsu).

Impor

```

import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.MultipleFileUpload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;
import java.util.ArrayList;

```

```
import java.util.Arrays;
```

Kode

```
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    MultipleFileUpload xfer = xfer_mgr.uploadDirectory(bucket_name,
        key_prefix, new File(dir_path), recursive);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Lihat [Tunggu Transfer untuk Menyelesaikan](#) untuk informasi tentang menggunakan `waitForCompletion` untuk berhasil menyelesaikan transfer sebelum memanggil `TransferManager.shutdownNow` metode. Sambil menunggu transfer selesai, Anda dapat memilih atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat [Dapatkan Status Transfer dan Kemajuan](#) untuk informasi lebih lanjut.

Parameter [MultipleFileUpload](#) objek yang dikembalikan oleh `uploadFileList` dapat digunakan untuk query negara transfer atau kemajuan. Lihat [Poll Kemajuan Transfer Saat Ini](#) dan [Dapatkan Transfer Progress dengan ProgressListener](#) untuk informasi lebih lanjut.

Anda juga dapat menggunakan `MultipleFileUpload`'s `getSubTransfers` metode untuk mendapatkan `IndividualUpload` objek untuk setiap file yang ditransfer. Untuk informasi selengkapnya, lihat [Dapatkan Kemajuan Subtransfer](#).

Lihat [Lengkapi Contoh](#) di GitHub.

Mengunduh File atau Direktori

Menggunakan `TransferManager` kelas untuk men-download salah satu file (Amazon S3 objek) atau direktori (sebuah Amazon S3 nama bucket diikuti oleh awalan objek) dari Amazon S3.

Topik

- [Mengunduh File Tunggal](#)

- [Unduh Direktori](#)

Mengunduh File Tunggal

Gunakan `TransferManager.download` metode, menyediakan Amazon S3 nama bucket yang berisi objek yang ingin Anda download, kunci (objek) nama, dan [Berkas](#) objek yang mewakili file untuk membuat pada sistem lokal Anda.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.Download;
import com.amazonaws.services.s3.transfer.MultipleFileDownload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;

import java.io.File;
```

Kode

```
File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Download xfer = xfer_mgr.download(bucket_name, key_name, f);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Lihat [Tunggu Transfer untuk Menyelesaikan](#) untuk informasi tentang menggunakan `waitForCompletion` untuk berhasil menyelesaikan transfer sebelum memanggil `TransferManager.shutdownNow` metode. Sambil menunggu transfer selesai, Anda dapat memilih atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat [Dapatkan Status Transfer dan Kemajuan](#) untuk informasi lebih lanjut.

Lihat [Lengkapi Contoh](#) di GitHub.

Unduh Direktori

Untuk mengunduh satu set file yang berbagi key prefix umum (analog dengan direktori pada sistem file) dari Amazon S3, gunakan `TransferManager.downloadDirectory` metode. Metode ini mengambil Amazon S3 nama bucket yang berisi objek yang ingin Anda download, awalan objek bersama oleh semua objek, dan [Berkas](#) objek yang mewakili direktori untuk men-download file ke dalam pada sistem lokal Anda. Jika direktori bernama belum ada, maka akan dibuat.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.Download;
import com.amazonaws.services.s3.transfer.MultipleFileDownload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;

import java.io.File;
```

Kode

```
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();

try {
    MultipleFileDownload xfer = xfer_mgr.downloadDirectory(
        bucket_name, key_prefix, new File(dir_path));
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Lihat [Tunggu Transfer untuk Menyelesaikan](#) untuk informasi tentang menggunakan `waitForCompletion` untuk berhasil menyelesaikan transfer sebelum memanggil `TransferManager.shutdownNow` metode. Sambil menunggu transfer selesai, Anda dapat memilih atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat [Dapatkan Status Transfer dan Kemajuan](#) untuk informasi lebih lanjut.

Lihat [Lengkapi Contoh](#) di GitHub.

Salin Objek

Untuk menyalin objek dari satu bucket S3 ke yang lain, gunakan `TransferManager` `copy` metode.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
```

Kode

```
System.out.println("Copying s3 object: " + from_key);
System.out.println("    from bucket: " + from_bucket);
System.out.println("    to s3 object: " + to_key);
System.out.println("    in bucket: " + to_bucket);

TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Copy xfer = xfer_mgr.copy(from_bucket, from_key, to_bucket, to_key);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Lihat [Lengkapi Contoh](#) di GitHub.

Tunggu Transfer untuk Menyelesaikan

Jika aplikasi Anda (atau thread) dapat memblokir sampai transfer selesai, Anda dapat menggunakan [transfer](#) `waitForCompletion` metode untuk memblokir sampai transfer selesai atau pengecualian terjadi.

```
try {
    xfer.waitForCompletion();
} catch (AmazonServiceException e) {
```

```

    System.err.println("Amazon service error: " + e.getMessage());
    System.exit(1);
} catch (AmazonClientException e) {
    System.err.println("Amazon client error: " + e.getMessage());
    System.exit(1);
} catch (InterruptedException e) {
    System.err.println("Transfer interrupted: " + e.getMessage());
    System.exit(1);
}

```

Anda mendapatkan kemajuan transfer jika Anda memilih untuk `waitForCompletion`, menerapkan mekanisme pemungutan suara pada thread terpisah, atau menerima pembaruan kemajuan secara asinkron menggunakan [ProgressListener](#).

Lihat [Lengkapi Contoh](#) di GitHub.

Dapatkan Status Transfer dan Kemajuan

Masing-masing kelas yang dikembalikan oleh `TransferManager.upload*`, `download*`, dan `copy` metode mengembalikan sebuah contoh dari salah satu kelas berikut, tergantung pada apakah itu tunggal-file atau operasi multiple-file.

Kelas	Dikembalikan oleh
Salin	copy
Unduh	download
MultipleFileDownload	downloadDirectory
Unggah	upload
MultipleFileUpload	uploadFileList , uploadDirectory

Semua kelas ini menerapkan [transfer](#) antarmuka. `Transfer` menyediakan metode yang berguna untuk mendapatkan kemajuan transfer, jeda atau melanjutkan transfer, dan mendapatkan status transfer saat ini atau akhir.

Topik

- [Poll Kemajuan Transfer Saat Ini](#)
- [Dapatkan Transfer Progress dengan ProgressListener](#)
- [Dapatkan Kemajuan Subtransfer](#)

Poll Kemajuan Transfer Saat Ini

Lingkaran ini mencetak kemajuan transfer, memeriksa kemajuan saat ini saat berjalan dan, ketika selesai, mencetak keadaan akhir.

Impor

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;

import java.io.File;
import java.util.ArrayList;
import java.util.Collection;
```

Kode

```
// print the transfer's human-readable description
System.out.println(xfer.getDescription());
// print an empty progress bar...
printProgressBar(0.0);
// update the progress bar while the xfer is ongoing.
do {
    try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        return;
    }
    // Note: so_far and total aren't used, they're just for
    // documentation purposes.
    TransferProgress progress = xfer.getProgress();
    long so_far = progress.getBytesTransferred();
    long total = progress.getTotalBytesToTransfer();
    double pct = progress.getPercentTransferred();
```



```

    eraseProgressBar();
    printProgressBar(pct);
} while (xfer.isDone() == false);
// print the final state of the transfer.
TransferState xfer_state = xfer.getState();
System.out.println(": " + xfer_state);

```

Lihat [Lengkapi Contoh](#) di GitHub.

Dapatkan Transfer Progress dengan ProgressListener

Anda dapat melampirkan [ProgressListener](#) ke transfer apa pun dengan menggunakan [transfer](#) Antarmuka `addProgressListener` metode.

SEBUAH [ProgressListener](#) hanya membutuhkan satu metode, `progressChanged`, yang mengambil [ProgressEvent](#) objek. Anda dapat menggunakan objek untuk mendapatkan total byte operasi dengan memanggil `getBytes` metode, dan jumlah byte ditransfer sejauh ini dengan memanggil `getBytesTransferred`.

Impor

```

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;

import java.io.File;
import java.util.ArrayList;
import java.util.Collection;

```

Kode

```

File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Upload u = xfer_mgr.upload(bucket_name, key_name, f);
    // print an empty progress bar...
    printProgressBar(0.0);
    u.addProgressListener(new ProgressListener() {
        public void progressChanged(ProgressEvent e) {
            double pct = e.getBytesTransferred() * 100.0 / e.getBytes();

```

```

        eraseProgressBar();
        printProgressBar(pct);
    }
});
// block with Transfer.waitForCompletion()
XferMgrProgress.waitForCompletion(u);
// print the final state of the transfer.
TransferState xfer_state = u.getState();
System.out.println(": " + xfer_state);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();

```

Lihat [Lengkapi Contoh](#) di GitHub.

Dapatkan Kemajuan Subtransfer

Parameter [MultipleFileUpload](#) kelas dapat mengembalikan informasi tentang subtransfernya dengan memanggil `nyagetSubTransfers` metode. Ia mengembalikan sebuah tidak dapat dimodifikasi [Koleksi](#) dari [Unggah](#) objek yang menyediakan status transfer individu dan kemajuan setiap sub-transfer.

Impor

```

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;

import java.io.File;
import java.util.ArrayList;
import java.util.Collection;

```

Kode

```

Collection<? extends Upload> sub_xfers = new ArrayList<Upload>();
sub_xfers = multi_upload.getSubTransfers();

do {

```

```
System.out.println("\nSubtransfer progress:\n");
for (Upload u : sub_xfers) {
    System.out.println(" " + u.getDescription());
    if (u.isDone()) {
        TransferState xfer_state = u.getState();
        System.out.println(" " + xfer_state);
    } else {
        TransferProgress progress = u.getProgress();
        double pct = progress.getPercentTransferred();
        printProgressBar(pct);
        System.out.println();
    }
}

// wait a bit before the next update.
try {
    Thread.sleep(200);
} catch (InterruptedException e) {
    return;
}

} while (multi_upload.isDone() == false);
// print the final state of the transfer.
TransferState xfer_state = multi_upload.getState();
System.out.println("\nMultipleFileUpload " + xfer_state);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Info Selengkapnya

- [Kunci objek](#) di Amazon Simple Storage Service Panduan Pengguna

Mengonfigurasi Amazon S3 Bucket sebagai Situs Web

Anda dapat mengonfigurasi Amazon S3 bucket untuk berperilaku sebagai situs web. Untuk melakukan ini, Anda perlu mengatur konfigurasi situs webnya.

Note

Contoh-contoh kode ini berasumsi bahwa Anda memahami materi di [Menggunakan AWS SDK for Java](#) dan telah dikonfigurasi default AWS kredensi menggunakan informasi di [Mengatur AWS Kredensial dan Wilayah untuk Pembangunan](#).

Mengatur Konfigurasi Situs Web Bucket

Untuk mengatur Amazon S3 konfigurasi situs bucket, hubungi `AmazonS3.setWebsiteConfiguration` metode dengan nama bucket untuk mengatur konfigurasi untuk, dan [BucketWebsiteConfiguration](#) objek yang berisi konfigurasi website bucket.

Mengatur dokumen indeks dibutuhkan; Semua parameter lainnya bersifat opsional.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketWebsiteConfiguration;
```

Kode

```
String bucket_name, String index_doc, String error_doc) {
    BucketWebsiteConfiguration website_config = null;

    if (index_doc == null) {
        website_config = new BucketWebsiteConfiguration();
    } else if (error_doc == null) {
        website_config = new BucketWebsiteConfiguration(index_doc);
    } else {
        website_config = new BucketWebsiteConfiguration(index_doc, error_doc);
    }

    final AmazonS3 s3 =
        AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
    try {
        s3.setBucketWebsiteConfiguration(bucket_name, website_config);
    } catch (AmazonServiceException e) {
        System.out.format(
            "Failed to set website configuration for bucket '%s'!\n",
            bucket_name);
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Note

Menetapkan konfigurasi situs web tidak memodifikasi izin akses untuk bucket Anda. Untuk membuat file Anda terlihat di web, Anda juga perlu mengatur kebijakan bucket yang memungkinkan akses baca publik ke file dalam bucket. Untuk informasi selengkapnya, lihat [Mengelola Akses ke Amazon S3](#) [member menggunakan kebijakan bucket](#).

Lihat [Lengkapi Contoh](#) di GitHub.

Dapatkan Konfigurasi Situs Web Bucket

Untuk mendapatkan Amazon S3 konfigurasi situs bucket, hubungi `AmazonS3.getWebsiteConfiguration` metode dengan nama bucket untuk mengambil konfigurasi untuk.

Konfigurasi akan dikembalikan sebagai [BucketWebsiteConfiguration](#) objek. Jika tidak ada konfigurasi situs web untuk bucket, maka `null` akan dikembalikan

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketWebsiteConfiguration;
```

Kode

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    BucketWebsiteConfiguration config =
        s3.getBucketWebsiteConfiguration(bucket_name);
    if (config == null) {
        System.out.println("No website configuration found!");
    } else {
        System.out.format("Index document: %s\n",
            config.getIndexDocumentSuffix());
        System.out.format("Error document: %s\n",
```

```
        config.getErrorDocument());
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.out.println("Failed to get website configuration!");
    System.exit(1);
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Menghapus Konfigurasi Situs Web Bucket

Untuk menghapus Amazon S3 konfigurasi situs bucket, hubungi `AmazonS3.deleteWebsiteConfiguration` dengan nama bucket untuk menghapus konfigurasi dari.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

Kode

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.deleteBucketWebsiteConfiguration(bucket_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.out.println("Failed to delete website configuration!");
    System.exit(1);
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Informasi Selengkapnya

- [Situs web Bucket PUT](#) di Amazon S3 Referensi API
- [DAPATKAN situs web Bucket](#) di Amazon S3 Referensi API

- [BERHAPUS situs web Bucket](#) di Amazon S3 Referensi API

Gunakan Amazon S3 Enkripsi sisi klien

Mengenkripsi data menggunakan Amazon S3 enkripsi klien adalah salah satu cara Anda dapat memberikan lapisan perlindungan tambahan untuk informasi sensitif yang Anda simpan di Amazon S3. Contoh dalam bagian ini menunjukkan cara membuat dan mengonfigurasi Amazon S3 enkripsi klien untuk aplikasi Anda.

Jika Anda baru mengenal kriptografi, lihat [Dasar Kriptografi](#) di AWS Panduan Pengembang KMS untuk gambaran dasar istilah kriptografi dan algoritma. Untuk informasi tentang dukungan kriptografi di semua AWS SDK, lihat [AWS Support SDK untuk Amazon S3 Enkripsi sisi klien](#) di Amazon Web Services Referensi Umum.

Note

Contoh-contoh kode ini berasumsi bahwa Anda memahami materi di [Menggunakan AWS SDK for Java](#) dan telah dikonfigurasi default AWS kredensi menggunakan informasi di [Mengatur AWS Kredensial dan Wilayah untuk Pembangunan](#).

Jika Anda menggunakan versi 1.11.836 atau versi sebelumnya AWS SDK for Java, lihat [Amazon S3 Migrasi Klien](#) untuk informasi tentang migrasi aplikasi Anda ke versi yang lebih baru. Jika Anda tidak dapat bermigrasi, lihat [contoh lengkap ini](#) di GitHub.

Jika tidak, jika Anda menggunakan versi 1.11.837 atau yang lebih baru dari AWS SDK for Java, jelajahi contoh topik yang tercantum di bawah ini untuk digunakan Amazon S3 enkripsi di sisi klien.

Topik

- [Amazon S3 enkripsi sisi klien](#)
- [Amazon S3 enkripsi sisi klien dengan AWS Kunci terkelola KMS](#)

Amazon S3 enkripsi sisi klien

Contoh berikut menggunakan [AmazonS3EncryptionClientV2Builder](#) kelas untuk membuat Amazon S3 enkripsi sisi klien diaktifkan. Setelah diaktifkan, objek apa pun yang Anda unggah Amazon S3 menggunakan klien ini akan dienkripsi. Setiap benda yang Anda dapatkan dari Amazon S3 menggunakan klien ini akan secara otomatis didekripsi.

Note

Contoh berikut menunjukkan menggunakan Amazon S3 enkripsi sisi klien dengan kunci master klien. Untuk mempelajari cara menggunakan enkripsi dengan AWS Kunci terkelola KMS, lihat [Amazon S3 enkripsi sisi klien dengan AWS Kunci terkelola KMS](#).

Anda dapat memilih dari dua mode enkripsi saat mengaktifkan sisi klien Amazon S3 enkripsi: ketat dikonfirmasi atau diautentikasi. Bagian berikut menunjukkan cara mengaktifkan setiap jenis. Untuk mempelajari algoritma yang digunakan setiap mode, lihat [CryptoMode](#) definisi.

Impor yang diperlukan

Impor kelas berikut untuk contoh-contoh ini.

Impor

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3EncryptionClientV2Builder;
import com.amazonaws.services.s3.AmazonS3EncryptionV2;
import com.amazonaws.services.s3.model.CryptoConfigurationV2;
import com.amazonaws.services.s3.model.CryptoMode;
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.StaticEncryptionMaterialsProvider;
```

Enkripsi terautentikasi

Enkripsi otentikasi yang ketat adalah mode default jika tidak `CryptoMode` ditentukan.

Untuk secara eksplisit mengaktifkan mode ini, tentukan `StrictAuthenticatedEncryptionvalue` dalam `withCryptoConfiguration` metode.

Note

Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan yang terbaru [Goyang Puri jar](#) file di classpath aplikasi Anda.

Kode


```
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
CryptoConfigurationV2().withCryptoMode((CryptoMode.StrictAuthenticatedEncryption)))
    .withEncryptionMaterialsProvider(new StaticEncryptionMaterialsProvider(new
EncryptionMaterials(secretKey)))
    .build();

s3Encryption.putObject(bucket_name, ENCRYPTED_KEY2, "This is the 2nd content to
encrypt");
```

Mode enkripsi

Saat Anda menggunakan `AuthenticatedEncryptionmode`, algoritma pembungkus kunci ditingkatkan diterapkan selama enkripsi. Saat mendekripsi dalam mode ini, algoritma dapat memverifikasi integritas objek yang didekripsi dan membuang pengecualian jika cek gagal. Untuk detail lebih lanjut tentang cara kerja enkripsi terautentikasi, lihat [Amazon S3Enkripsi terautentikasi](#) posting blog.

Note

Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan yang terbaru [Goyang Puri jarfile](#) di classpath aplikasi Anda.

Untuk mengaktifkan mode ini, tentukan `AuthenticatedEncryptionvalue` dalam `withCryptoConfiguration` metode.

Kode

```
AmazonS3EncryptionV2 s3EncryptionClientV2 =
AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .withClientConfiguration(new ClientConfiguration())
    .withCryptoConfiguration(new
CryptoConfigurationV2().withCryptoMode(CryptoMode.AuthenticatedEncryption))
    .withEncryptionMaterialsProvider(new StaticEncryptionMaterialsProvider(new
EncryptionMaterials(secretKey)))
    .build();
```

```
s3EncryptionClientV2.putObject(bucket_name, ENCRYPTED_KEY1, "This is the 1st content to encrypt");
```

Amazon S3enkripsi sisi klien denganAWSKunci terkelola KMS

Contoh berikut menggunakan [AmazonS3EncryptionClientV2Builder](#) kelas untuk membuat Amazon S3enkripsi di sisi klien diaktifkan. Setelah dikonfigurasi, objek apa pun yang Anda unggah Amazon S3 menggunakan klien ini akan dienkripsi. Setiap benda yang Anda dapatkan dari Amazon S3 menggunakan klien ini secara otomatis didekripsi.

Note

Contoh berikut menunjukkan cara menggunakan Amazon S3enkripsi sisi klien dengan AWSKMS kunci yang dikelola. Untuk mempelajari cara menggunakan enkripsi dengan kunci Anda sendiri, lihat [Amazon S3enkripsi sisi klien](#).

Anda dapat memilih dari dua mode enkripsi saat mengaktifkan sisi klien Amazon S3enkripsi: ketat dikonfirmasi atau diautentikasi. Bagian berikut menunjukkan cara mengaktifkan setiap jenis. Untuk mempelajari algoritma yang digunakan setiap mode, lihat [CryptoMode](#) definisi.

Impor yang diperlukan

Impor kelas berikut untuk contoh-contoh ini.

Impor

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.GenerateDataKeyRequest;
import com.amazonaws.services.kms.model.GenerateDataKeyResult;
import com.amazonaws.services.s3.AmazonS3EncryptionClientV2Builder;
import com.amazonaws.services.s3.AmazonS3EncryptionV2;
import com.amazonaws.services.s3.model.CryptoConfigurationV2;
import com.amazonaws.services.s3.model.CryptoMode;
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.KMSEncryptionMaterialsProvider;
```

enkripsi terautentikasi

Enkripsi otentikasi yang ketat adalah mode default jika tidak `CryptoMode` ditentukan.

Untuk secara eksplisit mengaktifkan mode ini, tentukan `StrictAuthenticatedEncryption` value dalam `withCryptoConfiguration` metode.

Note

Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan yang terbaru [Goyang Puri jarfile](#) di classpath aplikasi Anda.

Kode

```
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
    CryptoConfigurationV2().withCryptoMode((CryptoMode.StrictAuthenticatedEncryption)))
    .withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))
    .build();

s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
with a key created in the {console}");
System.out.println(s3Encryption.getObjectAsString(bucket_name, ENCRYPTED_KEY3));
```

Panggil `putObject` metode pada metode Amazon S3 enkripsi klien untuk meng-upload objek.

Kode

```
s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
with a key created in the {console}");
```

Anda dapat mengambil objek menggunakan klien yang sama. Contoh ini memanggil `getObjectAsString` metode untuk mengambil string yang disimpan.

Kode

```
System.out.println(s3Encryption.getObjectAsString(bucket_name, ENCRYPTED_KEY3));
```

Mode enkripsi terautentik

Saat Anda menggunakan `AuthenticatedEncryptionmode`, algoritma pembungkus kunci ditingkatkan diterapkan selama enkripsi. Saat mendekripsi dalam mode ini, algoritma dapat memverifikasi integritas objek yang didekripsi dan membuang pengecualian jika cek gagal. Untuk detail lebih lanjut tentang cara kerja enkripsi terautentikasi, lihat [Amazon S3 Enkripsi terautentikasi sisi klien](#) posting blog.

Note

Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan yang terbaru [Goyang Puri jar](#) file di classpath aplikasi Anda.

Untuk mengaktifkan mode ini, tentukan `AuthenticatedEncryptionvalue` dalam `withCryptoConfiguration` metode.

Kode

```
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
CryptoConfigurationV2().withCryptoMode((CryptoMode.AuthenticatedEncryption)))
    .withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))
    .build();
```

Mengonfigurasi AWS KMS klien

Parameter Amazon S3 enkripsi klien menciptakan AWS KMS klien secara default, kecuali salah satu secara eksplisit ditentukan.

Untuk mengatur wilayah untuk ini dibuat secara otomatis AWS KMS klien, mengatur `awsKmsRegion`.

Kode

```
Region kmsRegion = Region.getRegion(Regions.AP_NORTHEAST_1);

AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
CryptoConfigurationV2().withAwsKmsRegion(kmsRegion))
```

```
.withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))  
.build();
```

Atau, Anda dapat menggunakan sendiri AWS KMS klien untuk menginisialisasi klien enkripsi.

Kode

```
AWSKMS kmsClient = AWSKMSClientBuilder.standard()  
    .withRegion(Regions.US_WEST_2);  
    .build();  
  
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()  
    .withRegion(Regions.US_WEST_2)  
    .withKmsClient(kmsClient)  
    .withCryptoConfiguration(new  
    CryptoConfigurationV2().withCryptoMode((CryptoMode.AuthenticatedEncryption)))  
    .withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))  
    .build();
```

Amazon SQS Contoh Menggunakan AWS SDK for Java

Bagian ini menyediakan contoh pemrograman [Amazon SQS](#) menggunakan [AWS SDK for Java](#).

Note

Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parameter [kode contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk membangun dan menjalankan.

Topik

- [Bekerja dengan Amazon SQS Antrean pesan](#)
- [Mengirim, Menerima, dan Menghapus Amazon SQS Pesan](#)
- [Mengaktifkan Jajak pendapat panjang Amazon SQS Antrean](#)
- [Mengatur Timeout Visibilitas di Amazon SQS](#)
- [Menggunakan Antrian Surat Mati di Amazon SQS](#)

Bekerja dengan Amazon SQS Antrean pesan

SEBUAH Antrean pesan adalah wadah logis yang digunakan untuk mengirim pesan andal Amazon SQS. Ada dua tipe antrian: standar dan pertama-dalam, pertama-keluar (FIFO). Untuk mempelajari lebih lanjut tentang antrian dan perbedaan antara jenis ini, lihat [Amazon SQS Panduan Pengembang](#).

Topik ini menjelaskan cara membuat, membuat daftar, menghapus, dan mendapatkan URL Amazon SQS antrian dengan menggunakan AWS SDK for Java.

Membuat Antrean

Gunakan klien `AmazonSqs.createQueue` metode, menyediakan [CreateQueueRequest](#) objek yang menggambarkan parameter antrian.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
```

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
CreateQueueRequest create_request = new CreateQueueRequest(QueueName)
    .addAttributesEntry("DelaySeconds", "60")
    .addAttributesEntry("MessageRetentionPeriod", "86400");

try {
    sqs.createQueue(create_request);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Anda dapat menggunakan bentuk sederhana `createQueue`, yang hanya membutuhkan nama antrian, untuk membuat antrian standar.

```
sqs.createQueue("MyQueue" + new Date().getTime());
```

Lihat [Lengkapi Contoh](#) di GitHub.

Antrean

Untuk membuat daftar Amazon SQS antrian untuk akun Anda, hubungi klien `AmazonSqs.listQueues` metode.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ListQueuesResult;
```

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
ListQueuesResult lq_result = sqs.listQueues();
System.out.println("Your SQS Queue URLs:");
for (String url : lq_result.getQueueUrls()) {
    System.out.println(url);
}
```

Menggunakan `listQueues` overload tanpa parameter kembali Antrean. Anda dapat menyaring hasil yang dikembalikan dengan melewatkannya `ListQueuesRequest` objek.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ListQueuesRequest;
```

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
String name_prefix = "Queue";
lq_result = sqs.listQueues(new ListQueuesRequest(name_prefix));
System.out.println("Queue URLs with prefix: " + name_prefix);
for (String url : lq_result.getQueueUrls()) {
    System.out.println(url);
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Mendapatkan URL untuk Antrian

Hubungi klien AmazonSqs `getQueueUrl` metode.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;  
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
String queue_url = sqs.getQueueUrl(QueueName).getQueueUrl();
```

Lihat [Lengkapi Contoh](#) di GitHub.

Menghapus Antrean

Berikan antrean [URL](#) ke klien AmazonSqs `deleteQueue` metode.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;  
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
sqs.deleteQueue(queue_url);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Info Selengkapnya

- [Bagaimana Amazon SQS Antrean](#) di Amazon SQS Panduan Pengembang
- [CreateQueue](#) di Amazon SQS Referensi API
- [GetQueueUrl](#) di Amazon SQS Referensi API
- [ListQueues](#) di Amazon SQS Referensi API

- [DeleteQueues](#) di Amazon SQS Referensi API

Mengirim, Menerima, dan Menghapus Amazon SQS Pesan

Topik ini menjelaskan cara mengirim, menerima, dan menghapus Amazon SQS pesan. Pesan selalu dikirimkan menggunakan [Antrean SQS](#).

Mengirim pesan

Menambahkan satu pesan ke Amazon SQS antrian dengan memanggil klien `AmazonSqs.sendMessage` metode. Berikan [SendMessageRequest](#) objek yang berisi antrian [URL](#), badan pesan, dan nilai penundaan opsional (dalam detik).

Impor

```
import com.amazonaws.services.sqs.AmazonSqs;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.SendMessageRequest;
```

Kode

```
SendMessageRequest send_msg_request = new SendMessageRequest()
    .withQueueUrl(queueUrl)
    .withMessageBody("hello world")
    .withDelaySeconds(5);
sqs.sendMessage(send_msg_request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Kirim Beberapa Pesan Sekaligus

Anda dapat mengirim lebih dari satu pesan dalam satu permintaan. Untuk mengirim beberapa pesan, gunakan klien `AmazonSqs.sendMessageBatch` metode, yang mengambil [SendMessageBatchRequest](#) berisi URL antrian dan daftar pesan (masing-masing [SendMessageBatchRequestEntry](#)) untuk mengirim. Anda juga dapat menetapkan nilai penundaan opsional per pesan.

Impor

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
```

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequestEntry;
```

Kode

```
SendMessageBatchRequest send_batch_request = new SendMessageBatchRequest()
    .withQueueUrl(queueUrl)
    .withEntries(
        new SendMessageBatchRequestEntry(
            "msg_1", "Hello from message 1"),
        new SendMessageBatchRequestEntry(
            "msg_2", "Hello from message 2")
            .withDelaySeconds(10));
sqs.sendMessageBatch(send_batch_request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Menerima Pesan

Mengambil setiap pesan yang saat ini dalam antrian dengan memanggil klien `AmazonSqs.receiveMessage` metode, melewatinya URL antrian ini. Pesan dikembalikan sebagai daftar [Pesan](#) benda.

Impor

```
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
```

Kode

```
List<Message> messages = sqs.receiveMessage(queueUrl).getMessages();
```

Hapus Pesan setelah Tanda Terima

Setelah menerima pesan dan memproses isinya, hapus pesan dari antrian dengan mengirimkan pegangan tanda terima pesan dan URL antrian ke klien `AmazonSQS.deleteMessage` metode.

Kode

```
for (Message m : messages) {
    sqs.deleteMessage(queueUrl, m.getReceiptHandle());
}
```

```
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Info Selengkapnya

- [Bagaimana Amazon SQS Antrean Kerja](#) di Amazon SQS Panduan Pengembang
- [SendMessage](#) di Amazon SQS Referensi API
- [SendMessageBatch](#) di Amazon SQS Referensi API
- [ReceiveMessage](#) di Amazon SQS Referensi API
- [DeleteMessage](#) di Amazon SQS Referensi API

Mengaktifkan Jajak pendapat panjang Amazon SQS Antrean

Amazon SQS menggunakan jajak pendapat secara default, menanyakan hanya subset dari server—berdasarkan distribusi acak tertimbang—untuk menentukan apakah ada pesan yang tersedia untuk dimasukkan dalam respons.

Jajak pendapat panjang membantu mengurangi biaya penggunaan Amazon SQS dengan mengurangi jumlah tanggapan kosong ketika tidak ada pesan yang tersedia untuk kembali sebagai balasan `ReceiveMessage` permintaan dikirim ke Amazon SQS antrian dan menghilangkan tanggapan kosong palsu.

Note

Anda dapat mengatur frekuensi polling panjang dari 1-20 detik.

Mengaktifkan Polling Panjang saat Membuat Antrian

Untuk mengaktifkan polling panjang saat membuat Amazon SQS antrian, mengatur `ReceiveMessageWaitTimeSeconds` atribut pada [CreateQueueRequest](#) objek sebelum memanggil kelas `AmazonSqs.createQueue` metode.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;  
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

```
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
```

Kode

```
final AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();

// Enable long polling when creating a queue
CreateQueueRequest create_request = new CreateQueueRequest()
    .withQueueName(queue_name)
    .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");

try {
    sqs.createQueue(create_request);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Mengaktifkan Polling Panjang pada Antrian yang Ada

Selain mengaktifkan polling panjang saat membuat antrian, Anda juga dapat mengaktifkannya pada antrian yang ada dengan

menyetel `ReceiveMessageWaitTimeSeconds` pada [SetQueueAttributesRequest](#) sebelum memanggil kelas `AmazonSqs.setQueueAttributes` metode.

Impor

```
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Kode

```
SetQueueAttributesRequest set_attrs_request = new SetQueueAttributesRequest()
    .withQueueUrl(queue_url)
    .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");
sqs.setQueueAttributes(set_attrs_request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Mengaktifkan Pemungutan Suara Panjang pada Tanda Terima Pesan

Anda dapat mengaktifkan polling panjang saat menerima pesan dengan mengatur waktu tunggu dalam hitungan detik pada [ReceiveMessageRequest](#) yang Anda suplai ke kelas `AmazonSqsreceiveMessage` metode.

Note

Anda harus memastikan bahwa AWS Batas waktu permintaan klien lebih besar dari waktu jajak pendapat panjang maksimum (20-an) sehingga Anda `receiveMessage` permintaan tidak habis saat menunggu acara jajak pendapat berikutnya!

Impor

```
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
```

Kode

```
ReceiveMessageRequest receive_request = new ReceiveMessageRequest()  
    .withQueueUrl(queue_url)  
    .withWaitTimeSeconds(20);  
sqs.receiveMessage(receive_request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Info Selengkapnya

- [Amazon SQS Jajak pendapat](#) di Amazon SQS Panduan Pengembang
- [CreateQueue](#) di Amazon SQS Referensi API
- [ReceiveMessage](#) di Amazon SQS Referensi API
- [setQueueAttributes](#) di Amazon SQS Referensi API

Mengatur Timeout Visibilitas di Amazon SQS

Ketika pesan diterima di Amazon SQS, tetap pada antrian sampai dihapus untuk memastikan penerimaan. Sebuah pesan yang diterima, tetapi tidak dihapus, akan tersedia dalam permintaan

berikutnya setelah diberikan batas waktu visibilitas untuk membantu mencegah pesan diterima lebih dari sekali sebelum dapat diproses dan dihapus.

Note

Saat menggunakan [Antrean standar](#), waktu tunggu visibilitas bukanlah jaminan untuk menerima pesan dua kali. Jika Anda menggunakan antrian standar, pastikan kode Anda dapat menangani kasus di mana pesan yang sama telah dikirimkan lebih dari satu kali.

Mengatur Timeout Visibilitas Pesan untuk Satu Pesan

Ketika Anda telah menerima pesan, Anda dapat memodifikasi batas waktu visibilitas dengan melewati pegangan tanda terima dalam [ChangeMessageVisibilityRequest](#) yang Anda lewati ke kelas `AmazonSqs` `changeMessageVisibility` metode.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();

// Get the receipt handle for the first message in the queue.
String receipt = sqs.receiveMessage(queue_url)
    .getMessages()
    .get(0)
    .getReceiptHandle();

sqs.changeMessageVisibility(queue_url, receipt, timeout);
```

Lihat [contoh lengkap](#) di GitHub.

Mengatur Timeout Visibilitas Pesan untuk Beberapa Pesan sekaligus

Untuk mengatur batas waktu tunggu visibilitas pesan untuk beberapa pesan sekaligus, buat daftar [ChangeMessageVisibilityBatchRequestEntry](#) objek, masing-masing berisi string ID unik dan pegangan tanda terima. Kemudian, lewati daftar ke `Amazon SQS` `client.changeMessageVisibilityBatch` metode.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ChangeMessageVisibilityBatchRequestEntry;
import java.util.ArrayList;
import java.util.List;
```

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();

List<ChangeMessageVisibilityBatchRequestEntry> entries =
    new ArrayList<ChangeMessageVisibilityBatchRequestEntry>();

entries.add(new ChangeMessageVisibilityBatchRequestEntry(
    "unique_id_msg1",
    sqs.receiveMessage(queue_url)
        .getMessages()
        .get(0)
        .getReceiptHandle())
    .withVisibilityTimeout(timeout));

entries.add(new ChangeMessageVisibilityBatchRequestEntry(
    "unique_id_msg2",
    sqs.receiveMessage(queue_url)
        .getMessages()
        .get(0)
        .getReceiptHandle())
    .withVisibilityTimeout(timeout + 200));

sqs.changeMessageVisibilityBatch(queue_url, entries);
```

Lihat [contoh lengkap](#) di GitHub.

Info Selengkapnya

- [Timeout Visibilitas](#) di dalam Amazon SQS Panduan Pengembang
- [setQueueAttributes](#) di dalam Amazon SQS Referensi API
- [GetQueueAttributes](#) di dalam Amazon SQS Referensi API
- [ReceiveMessage](#) di dalam Amazon SQS Referensi API

- [ChangeMessageVisibility](#) di dalam Amazon SQS Referensi API
- [ChangeMessageVisibilityBatch](#) di dalam Amazon SQS Referensi API

Menggunakan Antrian Surat Mati di Amazon SQS

Amazon SQS menyediakan dukungan untuk antrian surat mati. Antrian surat mati adalah antrian yang antrian (sumber) lainnya dapat menargetkan pesan yang tidak dapat berhasil diproses. Anda dapat menyisihkan dan mengisolasi pesan-pesan ini dalam antrian surat mati untuk menentukan mengapa pemrosesan mereka tidak berhasil.

Membuat Antrian Surat Mati

Antrian surat mati dibuat dengan cara yang sama seperti antrian biasa, tetapi memiliki batasan berikut:

- Antrian surat mati harus jenis antrian yang sama (FIFO atau standar) sebagai antrian sumber.
- Antrian surat mati harus dibuat menggunakan yang sama Akun AWS dan wilayah sebagai antrian sumber.

Di sini kita membuat dua identik Amazon SQS antrian, salah satunya akan berfungsi sebagai antrian surat mati:

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
```

Kode

```
final AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();

// Create source queue
try {
    sqs.createQueue(src_queue_name);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```



```
}

// Create dead-letter queue
try {
    sqs.createQueue(dl_queue_name);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Lihat [Lengkapi Contoh](#) di GitHub.

Menunjuk Antrian Surat Mati untuk Antrian Sumber

Untuk menunjuk antrian surat mati, Anda harus terlebih dahulu membuat kebijakan redrive, dan kemudian mengatur kebijakan dalam atribut antrian. Sebuah kebijakan redrive ditentukan dalam JSON, dan menentukan ARN antrian surat mati dan jumlah maksimum kali pesan dapat diterima dan tidak diproses sebelum dikirim ke antrian surat mati.

Untuk mengatur kebijakan redrive untuk antrian sumber Anda, hubungi kelas `AmazonSqsSetQueueAttributesMetode` dengan [SetQueueAttributesRequest](#) objek yang telah Anda atur `RedrivePolicy` atribut dengan kebijakan redrive JSON Anda.

Impor

```
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Kode

```
String dl_queue_url = sqs.getQueueUrl(dl_queue_name)
    .getQueueUrl();

GetQueueAttributesResult queue_attrs = sqs.getQueueAttributes(
    new GetQueueAttributesRequest(dl_queue_url)
    .withAttributeNames("QueueArn"));

String dl_queue_arn = queue_attrs.getAttributes().get("QueueArn");
```

```
// Set dead letter queue with redrive policy on source queue.
String src_queue_url = sqs.getQueueUrl(src_queue_name)
    .getQueueUrl();

SetQueueAttributesRequest request = new SetQueueAttributesRequest()
    .withQueueUrl(src_queue_url)
    .addAttributesEntry("RedrivePolicy",
        "{\"maxReceiveCount\": \"5\", \"deadLetterTargetArn\": \""
        + dl_queue_arn + "\"}");

sqs.setQueueAttributes(request);
```

Lihat [Lengkapi Contoh](#) di GitHub.

Info Selengkapnya

- [Menggunakan Amazon SQS Antrian Surat Mati](#) di Amazon SQS Panduan Pengembang
- [setQueueAttributes](#) di Amazon SQS Referensi API

Amazon SWF Contoh Menggunakan AWS SDK for Java

[Amazon SWF](#) adalah layanan manajemen alur kerja yang membantu pengembang membangun dan skala alur kerja terdistribusi yang dapat memiliki langkah-langkah paralel atau berurutan yang terdiri dari kegiatan, alur kerja anak atau bahkan [Lambda](#) tugas.

Ada dua cara untuk bekerja dengan Amazon SWF menggunakan AWS SDK for Java, dengan menggunakan SWF klien objek, atau dengan menggunakan AWS Flow Framework untuk Java. Parameter AWS Flow Framework untuk Java lebih sulit untuk mengkonfigurasi awalnya, karena membuat penggunaan berat anotasi dan bergantung pada perpustakaan tambahan seperti AspectJ dan Spring Framework. Namun, untuk proyek besar atau kompleks, Anda akan menghemat waktu pengkodean dengan menggunakan AWS Flow Framework untuk Java. Untuk informasi selengkapnya, lihat [AWS Flow Framework untuk Panduan Developer Java](#).

Bagian ini menyediakan contoh pemrograman Amazon SWF dengan menggunakan AWS SDK for Java klien secara langsung.

Topik

- [Dasar-dasar SWF](#)
- [Membangun Amazon SWF Aplikasi Sederhana](#)

- [Tugas Lambda](#)
- [Mematikan Kegiatan dan Alur Kerja Pekerja Anggun](#)
- [Mendaftarkan domain](#)
- [Daftar domain](#)

Dasar-dasar SWF

Ini adalah pola umum untuk bekerja dengan Amazon SWF menggunakan AWS SDK for Java. Hal ini dimaksudkan terutama untuk referensi. Untuk tutorial pengantar yang lebih lengkap, lihat [Membangun Sederhana Amazon SWF Aplikasi](#).

Dependensi

Dasar Amazon SWF aplikasi akan membutuhkan dependensi berikut, yang disertakan dengan AWS SDK for Java:

- aws-java-sdk-1.12.*.jar
- umum-logging-1.2.*.jar
- httpclient-4.3.*.jar
- httpcore-4.3.*.jar
- jackson-annotasi-2.12.*.jar
- jackson-inti-2.12.*.jar
- jackson-databind-2.12.*.jar
- joda-waktu-2.8.*.jar

Note

Nomor versi paket ini akan berbeda tergantung pada versi SDK yang Anda miliki, tetapi versi yang disertakan dengan SDK telah diuji untuk kompatibilitas, dan merupakan yang harus Anda gunakan.

AWS Flow Framework untuk aplikasi Java memerlukan pengaturan tambahan, dan dependensi tambahan. Lihat [AWS Flow Framework untuk Panduan Developer Java](#) Untuk informasi lebih lanjut tentang penggunaan framework.

Impor

Secara umum, Anda dapat menggunakan impor berikut untuk pengembangan kode:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Ini adalah praktik yang baik untuk mengimpor hanya kelas yang Anda butuhkan, namun. Anda mungkin akan berakhir menentukan kelas tertentu `com.amazonaws.services.simpleworkflow.model` Ruang kerja:

```
import com.amazonaws.services.simpleworkflow.model.PollForActivityTaskRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskCompletedRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskFailedRequest;
import com.amazonaws.services.simpleworkflow.model.TaskList;
```

Jika Anda menggunakan AWS Flow Framework untuk Java, Anda akan mengimpor kelas dari `com.amazonaws.services.simpleworkflow.flow` Ruang kerja. Misalnya:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.flow.ActivityWorker;
```

Note

Parameter AWS Flow Framework untuk Java memiliki persyaratan tambahan di luar basis AWS SDK for Java. Untuk informasi lebih lanjut, lihat [AWS Flow Framework untuk Panduan Developer Java](#).

Menggunakan kelas klien SWF

Antarmuka dasar Anda untuk Amazon SWF adalah melalui salah satu [AmazonSimpleWorkflowClient](#) atau [AmazonSimpleWorkflowAsyncClient](#) kelas. Perbedaan utama antara ini adalah bahwa `*AsyncClient` kelas kembali [Masa Depan](#) objek untuk pemrograman bersamaan (asinkron).

```
AmazonSimpleWorkflowClient swf = AmazonSimpleWorkflowClientBuilder.defaultClient();
```

Membangun Amazon SWF Aplikasi Sederhana

Topik ini akan memperkenalkan Anda ke [Amazon SWF](#) aplikasi pemrograman dengan AWS SDK for Java, sambil menyajikan beberapa konsep penting di sepanjang jalan.

Tentang contoh

Contoh proyek akan membuat alur kerja dengan satu aktivitas yang menerima data alur kerja yang dilewatkan melalui AWS cloud (Dalam tradisi HelloWorld, itu akan menjadi nama seseorang untuk disambut) dan kemudian mencetak salam sebagai tanggapan.

Meskipun ini tampak sangat sederhana di permukaan, Amazon SWF aplikasi terdiri dari sejumlah bagian yang bekerja sama:

- Domain, digunakan sebagai wadah logis untuk data eksekusi alur kerja Anda.
- Satu atau lebih alur kerja yang mewakili komponen kode yang menentukan urutan logis eksekusi aktivitas alur kerja dan alur kerja anak Anda.
- Pekerja alur kerja, juga dikenal sebagai penentu, yang melakukan polling untuk tugas keputusan dan menjadwalkan kegiatan atau alur kerja anak sebagai tanggapan.
- Satu atau lebih aktivitas, yang masing-masing mewakili unit kerja dalam alur kerja.
- Activity worker yang melakukan polling untuk tugas aktivitas dan menjalankan metode aktivitas sebagai respons.
- Satu atau lebih daftar tugas, yang antrian dikelola oleh Amazon SWF digunakan untuk mengeluarkan permintaan untuk alur kerja dan aktivitas pekerja. Tugas pada daftar tugas yang dimaksudkan untuk pekerja alur kerja disebut tugas keputusan. Yang dimaksudkan untuk pekerja aktivitas disebut tugas aktivitas.
- Starter alur kerja yang memulai eksekusi alur kerja Anda.

Di belakang layar, Amazon SWF mengatur pengoperasian komponen-komponen ini, mengoordinasikan alirannya dari AWS cloud, meneruskan data di antara mereka, menangani batas waktu dan pemberitahuan detak jantung, dan mencatat riwayat eksekusi alur kerja.

Prasyarat

Lingkungan pengembangan

Lingkungan pengembangan yang digunakan dalam tutorial ini terdiri dari:

Buat proyek SWF

1. Mulai proyek baru dengan Maven:

```
mvn archetype:generate -DartifactId=helloswf \
-DgroupId=aws.example.helloswf -DinteractiveMode=false
```

Ini akan membuat proyek baru dengan struktur proyek maven standar:

```
helloswf
### pom.xml
### src
### main
#   ### java
#       ### aws
#           ### example
#               ### helloswf
#                   ### App.java
### test
### ...
```

Anda dapat mengabaikan atau menghapus test direktori dan semua yang dikandungnya, kami tidak akan menggunakannya untuk tutorial ini. Anda juga dapat menghapus `App.java`, karena kami akan menggantinya dengan kelas baru.

2. Edit `pom.xml` file proyek dan tambahkan `aws-java-sdk-simpleworkflow` modul dengan menambahkan ketergantungan untuk itu di `<dependencies>` dalam blok.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-simpleworkflow</artifactId>
    <version>1.11.1000</version>
  </dependency>
</dependencies>
```

3. Pastikan Maven membangun proyek Anda dengan dukungan JDK 1.7+. Tambahkan yang berikut ke proyek Anda (baik sebelum atau sesudah `<dependencies>` blok) di `pom.xml`:

```
<build>
  <plugins>
```

```
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.6.1</version>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
  </configuration>
</plugin>
</plugins>
</build>
```

Kode proyek

Contoh proyek akan terdiri dari empat aplikasi terpisah, yang akan kita kunjungi satu per satu:

- HelloTypes.java -berisi data jenis domain, aktivitas, dan alur kerja proyek, yang dibagikan dengan komponen lainnya. Ini juga menangani pendaftaran jenis ini dengan SWF.
- ActivityWorker.java -berisi activity worker, yang melakukan polling untuk tugas aktivitas dan menjalankan aktivitas sebagai respons.
- WorkflowWorker.java --berisi pekerja alur kerja (decider), yang jajak pendapat untuk tugas keputusan dan jadwal kegiatan baru.
- WorkflowStarter.java -mengandung starter alur kerja, yang memulai eksekusi alur kerja baru, yang akan menyebabkan SWF mulai menghasilkan tugas keputusan dan alur kerja untuk dikonsumsi pekerja Anda.

Langkah-langkah umum untuk semua file sumber

Semua file yang Anda buat untuk rumah kelas Java Anda akan memiliki beberapa kesamaan. Untuk kepentingan waktu, langkah-langkah ini akan tersirat setiap kali Anda menambahkan file baru ke proyek:

1. Buat file di dalam `src/main/java/aws/example/helloswf/` direktori proyek.
2. Tambahkan package deklarasi ke awal setiap file untuk mendeklarasikan namespace. Contoh proyek menggunakan:

```
package aws.example.helloswf;
```


3. Tambahkan `import` deklarasi untuk [AmazonSimpleWorkflowClient](#) kelas dan untuk beberapa kelas di `com.amazonaws.services.simpleworkflow.model` namespace. Untuk menyederhanakan hal-hal, kita akan menggunakan:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Mendaftarkan jenis domain, alur kerja, dan jenis aktivitas

Kita akan mulai dengan membuat kelas `executable` baru, `HelloTypes.java`. File ini akan berisi data bersama yang perlu diketahui oleh berbagai bagian alur kerja Anda, seperti nama dan versi jenis aktivitas dan alur kerja, nama domain, dan nama daftar tugas.

1. Buka editor teks Anda dan buat file `HelloTypes.java`, tambahkan deklarasi paket dan impor sesuai dengan langkah-langkah [umum](#).
2. Deklarasikan `HelloTypes` kelas dan berikan nilai yang akan digunakan untuk aktivitas terdaftar dan jenis alur kerja Anda:

```
public static final String DOMAIN = "HelloDomain";
public static final String TASKLIST = "HelloTasklist";
public static final String WORKFLOW = "HelloWorkflow";
public static final String WORKFLOW_VERSION = "1.0";
public static final String ACTIVITY = "HelloActivity";
public static final String ACTIVITY_VERSION = "1.0";
```

Nilai-nilai ini akan digunakan di seluruh kode.

3. Setelah deklarasi `String`, membuat sebuah instance dari [AmazonSimpleWorkflowClient](#) kelas. Ini adalah antarmuka dasar untuk Amazon SWF metode yang disediakan oleh AWS SDK for Java.

```
private static final AmazonSimpleWorkflow swf =
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

Cuplikan sebelumnya mengasumsikan bahwa kredensial sementara dikaitkan dengan profil default. Jika Anda menggunakan profil yang berbeda, ubah kode di atas sebagai berikut dan ganti *profile_name dengan nama nama* profil yang sebenarnya.

```
private static final AmazonSimpleWorkflow swf =
    AmazonSimpleWorkflowClientBuilder
        .standard()
        .withCredentials(new ProfileCredentialsProvider("profile_name"))
        .withRegion(Regions.DEFAULT_REGION)
        .build();
```

4. Tambahkan fungsi baru untuk mendaftarkan domain SWF. Domain adalah wadah logis untuk sejumlah aktivitas SWF terkait dan jenis alur kerja. Komponen SWF hanya dapat berkomunikasi satu sama lain jika ada dalam domain yang sama.

```
try {
    System.out.println("** Registering the domain '" + DOMAIN + "'.");
    swf.registerDomain(new RegisterDomainRequest()
        .withName(DOMAIN)
        .withWorkflowExecutionRetentionPeriodInDays("1"));
} catch (DomainAlreadyExistsException e) {
    System.out.println("** Domain already exists!");
}
```

Ketika Anda mendaftarkan domain, Anda memberinya nama (set 1 - 256 karakter tidak termasuk :, / |, karakter kontrol atau string literal `arn') dan periode retensi, yang merupakan jumlah hari yang Amazon SWF akan menjaga data riwayat eksekusi alur kerja Anda setelah eksekusi alur kerja selesai. Batas penyimpanan eksekusi alur kerja maksimum adalah 90 hari. Lihat [RegisterDomainRequest](#) untuk informasi lebih lanjut.

Jika domain dengan nama itu sudah ada, akan [DomainAlreadyExistsException](#) dinaikkan. Karena kita tidak peduli jika domain telah dibuat, kita dapat mengabaikan pengecualian.

Note

Kode ini menunjukkan pola umum ketika bekerja dengan AWS SDK for Java metode, data untuk metode ini disediakan oleh kelas di `simpleworkflow.model` namespace, yang Anda instantiate dan mengisi menggunakan metode chainable. `@with*`

5. Tambahkan fungsi untuk mendaftarkan jenis aktivitas baru. Aktivitas mewakili unit kerja dalam alur kerja Anda.

```
try {
```

```

System.out.println("** Registering the activity type '" + ACTIVITY +
    "-" + ACTIVITY_VERSION + "'.");
swf.registerActivityType(new RegisterActivityTypeRequest()
    .withDomain(DOMAIN)
    .withName(ACTIVITY)
    .withVersion(ACTIVITY_VERSION)
    .withDefaultTaskList(new TaskList().withName(TASKLIST))
    .withDefaultTaskScheduleToStartTimeout("30")
    .withDefaultTaskStartToCloseTimeout("600")
    .withDefaultTaskScheduleToCloseTimeout("630")
    .withDefaultTaskHeartbeatTimeout("10"));
} catch (TypeAlreadyExistsException e) {
    System.out.println("** Activity type already exists!");
}

```

Jenis aktivitas diidentifikasi berdasarkan nama dan versi, yang digunakan untuk mengidentifikasi aktivitas secara unik dari orang lain di domain tempat terdaftar. Aktivitas juga berisi sejumlah parameter opsional, seperti daftar tugas default yang digunakan untuk menerima tugas dan data dari SWF dan sejumlah waktu tunggu berbeda yang dapat Anda gunakan untuk menempatkan batasan pada berapa lama bagian yang berbeda dari eksekusi aktivitas dapat dilakukan. Lihat [RegisterActivityTypeRequest](#) untuk informasi lebih lanjut.

Note

Semua nilai batas waktu ditentukan dalam hitungan detik. Lihat [Amazon SWF Timeout Types](#) untuk penjelasan lengkap tentang bagaimana batas waktu memengaruhi eksekusi alur kerja Anda.

Jika jenis aktivitas yang Anda coba daftarkan sudah ada, maka akan muncul.

[TypeAlreadyExistsException](#) Tambahkan fungsi untuk mendaftarkan jenis alur kerja baru. Alur kerja, juga dikenal sebagai penentu mewakili logika eksekusi alur kerja Anda.

+

```

try {
    System.out.println("** Registering the workflow type '" + WORKFLOW +
        "-" + WORKFLOW_VERSION + "'.");
    swf.registerWorkflowType(new RegisterWorkflowTypeRequest()
        .withDomain(DOMAIN)
        .withName(WORKFLOW)

```

```
        .withVersion(WORKFLOW_VERSION)
        .withDefaultChildPolicy(ChildPolicy.TERMINATE)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskStartToCloseTimeout("30"));
    } catch (TypeAlreadyExistsException e) {
        System.out.println("** Workflow type already exists!");
    }
```

+

Mirip dengan jenis aktivitas, jenis alur kerja diidentifikasi oleh nama dan versi dan juga memiliki batas waktu yang dapat dikonfigurasi. Lihat [RegisterWorkflowTypeRequest](#) untuk informasi lebih lanjut.

+

Jika jenis alur kerja yang Anda coba daftarkan sudah ada, sebuah akan [TypeAlreadyExistsException](#) dinaikkan. Akhirnya, membuat kelas executable dengan menyediakan main metode, yang akan mendaftarkan domain, jenis aktivitas, dan jenis alur kerja pada gilirannya:

+

```
registerDomain();
registerWorkflowType();
registerActivityType();
```

Anda dapat [membangun](#) dan [menjalankan](#) aplikasi sekarang untuk menjalankan skrip pendaftaran, atau melanjutkan pengkodean aktivitas dan pekerja alur kerja. Setelah domain, alur kerja, dan aktivitas telah didaftarkan, Anda tidak perlu menjalankannya lagi—jenis ini tetap ada sampai Anda mengusangnya sendiri.

Mengimplementasikan pekerja aktivitas

Suatu kegiatan adalah satuan dasar kerja dalam alur kerja. Alur kerja menyediakan logika, kegiatan penjadwalan yang akan dijalankan (atau tindakan lain yang harus diambil) dalam menanggapi tugas keputusan. Alur kerja tipikal biasanya terdiri dari sejumlah aktivitas yang dapat berjalan secara sinkron, asinkron, atau kombinasi keduanya.

Activity worker adalah sedikit kode yang melakukan polling untuk tugas aktivitas yang dihasilkan Amazon SWF sebagai respons terhadap keputusan alur kerja. Ketika menerima tugas aktivitas, ia menjalankan aktivitas yang sesuai dan mengembalikan respons sukses/kegagalan kembali ke alur kerja.

Kami akan menerapkan activity worker sederhana yang mendorong satu aktivitas.

1. Buka editor teks Anda dan buat `fileActivityWorker.java`, tambahkan deklarasi paket dan impor sesuai dengan langkah-langkah [umum](#).

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

2. Tambahkan `ActivityWorker` kelas ke file, dan berikan anggota data untuk memegang klien SWF yang akan kita gunakan untuk berinteraksi dengan Amazon SWF:

```
private static final AmazonSimpleWorkflow swf =
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

3. Tambahkan metode yang akan kita gunakan sebagai aktivitas:

```
private static String sayHello(String input) throws Throwable {
    return "Hello, " + input + "!";
}
```

Aktivitas hanya mengambil string, menggabungkannya menjadi salam dan mengembalikan hasilnya. Meskipun ada sedikit kemungkinan bahwa kegiatan ini akan menimbulkan pengecualian, itu ide yang baik untuk merancang kegiatan yang dapat meningkatkan kesalahan jika terjadi kesalahan.

4. Tambahkan `main` metode yang akan kita gunakan sebagai metode polling task activity. Kita akan memulainya dengan menambahkan beberapa kode untuk memilih daftar tugas untuk tugas aktivitas:

```
System.out.println("Polling for an activity task from the tasklist '"
    + HelloTypes.TASKLIST + "' in the domain '"
    + HelloTypes.DOMAIN + "'.");

ActivityTask task = swf.pollForActivityTask(
    new PollForActivityTaskRequest()
        .withDomain(HelloTypes.DOMAIN)
        .withTaskList(
            new TaskList().withName(HelloTypes.TASKLIST)));
```

```
String task_token = task.getTaskToken();
```

Aktivitas menerima tugas dari Amazon SWF dengan memanggil `pollForActivityTask` metode klien SWF, menentukan domain dan daftar tugas untuk digunakan dalam `passed-in`.

[PollForActivityTaskRequest](#)

Setelah tugas diterima, kita mengambil identifier unik untuk itu dengan memanggil metode tugas `getTaskToken`

5. Selanjutnya, tulis beberapa kode untuk memproses tugas yang masuk. Tambahkan yang berikut ke `main` metode Anda, tepat setelah kode yang polling untuk tugas dan mengambil token tugasnya.

```
if (task_token != null) {
    String result = null;
    Throwable error = null;

    try {
        System.out.println("Executing the activity task with input '" +
            task.getInput() + "'.");
        result = sayHello(task.getInput());
    } catch (Throwable th) {
        error = th;
    }

    if (error == null) {
        System.out.println("The activity task succeeded with result '" +
            result + "'.");
        swf.respondActivityTaskCompleted(
            new RespondActivityTaskCompletedRequest()
                .withTaskToken(task_token)
                .withResult(result));
    } else {
        System.out.println("The activity task failed with the error '" +
            error.getClass().getSimpleName() + "'.");
        swf.respondActivityTaskFailed(
            new RespondActivityTaskFailedRequest()
                .withTaskToken(task_token)
                .withReason(error.getClass().getSimpleName())
                .withDetails(error.getMessage()));
    }
}
```

```
}
```

Jika token tugas tidaknull, maka kita dapat mulai menjalankan metode aktivitas (`sayHello`), menyediakannya dengan data input yang dikirim bersama tugas.

Jika tugas berhasil (tidak ada kesalahan yang dihasilkan), maka pekerja merespons SWF dengan memanggil `respondActivityTaskCompleted` metode klien SWF dengan [RespondActivityTaskCompletedRequest](#) objek yang berisi token tugas dan data hasil aktivitas.

Di sisi lain, jika tugas gagal, maka kita merespons dengan memanggil `respondActivityTaskFailed` metode dengan [RespondActivityTaskFailedRequest](#) objek, meneruskannya token tugas dan informasi tentang kesalahan.

Note

Kegiatan ini tidak akan ditutup dengan anggun jika terbunuh. Meskipun berada di luar cakupan tutorial ini, implementasi alternatif dari activity worker ini disediakan dalam topik yang menyertainya, [Menutup Aktivitas dan Pekerja Alur Kerja](#) dengan Anggun.

Mengimplementasikan pekerja alur kerja

Logika alur kerja Anda berada dalam sepotong kode yang dikenal sebagai pekerja alur kerja. Pekerja alur kerja jajak pendapat untuk tugas-tugas keputusan yang dikirim oleh Amazon SWF dalam domain, dan pada daftar tugas default, bahwa jenis alur kerja terdaftar dengan.

Ketika pekerja alur kerja menerima tugas, itu membuat semacam keputusan (biasanya apakah akan menjadwalkan aktivitas baru atau tidak) dan mengambil tindakan yang tepat (seperti menjadwalkan aktivitas).

1. Buka editor teks Anda dan buat `fileWorkflowWorker.java`, tambahkan deklarasi paket dan impor sesuai dengan langkah-langkah [umum](#).
2. Tambahkan beberapa impor tambahan ke file:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
import java.util.ArrayList;
```

```
import java.util.List;
import java.util.UUID;
```

3. Mendeklarasikan `WorkflowWorker` kelas, dan membuat sebuah instance dari [AmazonSimpleWorkflowClient](#) kelas yang digunakan untuk mengakses metode SWF.

```
private static final AmazonSimpleWorkflow swf =
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

4. Tambahkan `main` metode. Metode loop terus menerus, polling untuk tugas-tugas keputusan menggunakan metode SWF klien. `pollForDecisionTask` [PollForDecisionTaskRequest](#) Memberikan detailnya.

```
PollForDecisionTaskRequest task_request =
    new PollForDecisionTaskRequest()
        .withDomain>HelloTypes.DOMAIN)
        .withTaskList(new TaskList().withName>HelloTypes.TASKLIST));

while (true) {
    System.out.println(
        "Polling for a decision task from the tasklist '" +
        HelloTypes.TASKLIST + "' in the domain '" +
        HelloTypes.DOMAIN + "'.");

    DecisionTask task = swf.pollForDecisionTask(task_request);

    String taskToken = task.getTaskToken();
    if (taskToken != null) {
        try {
            executeDecisionTask(taskToken, task.getEvents());
        } catch (Throwable th) {
            th.printStackTrace();
        }
    }
}
```

Setelah tugas diterima, kita memanggil `getTaskToken` metodenya, yang mengembalikan string yang dapat digunakan untuk mengidentifikasi tugas. Jika token yang dikembalikan tidak `null`, maka kita memprosesnya lebih lanjut dalam `executeDecisionTask` metode, meneruskannya token tugas dan daftar [HistoryEvent](#) objek yang dikirim dengan tugas.

5. Tambahkan `executeDecisionTask` metode, mengambil token tugas (`aString`) dan `HistoryEvent` daftar.

```
List<Decision> decisions = new ArrayList<Decision>();
String workflow_input = null;
int scheduled_activities = 0;
int open_activities = 0;
boolean activity_completed = false;
String result = null;
```

Kami juga menyiapkan beberapa anggota data untuk melacak hal-hal seperti:

- Daftar objek [Keputusan](#) yang digunakan untuk melaporkan hasil pengolahan tugas.
 - String untuk menahan masukan alur kerja yang disediakan oleh acara `WorkflowExecutionStarted` ""
 - hitungan aktivitas terjadwal dan terbuka (berjalan) untuk menghindari penjadwalan aktivitas yang sama ketika sudah dijadwalkan atau sedang berjalan.
 - boolean untuk menunjukkan bahwa aktivitas telah selesai.
 - String untuk menahan hasil aktivitas, untuk mengembalikannya sebagai hasil alur kerja kami.
6. Selanjutnya, tambahkan beberapa kode `executeDecisionTask` untuk memproses `HistoryEvent` objek yang dikirim dengan tugas, berdasarkan jenis acara yang dilaporkan oleh `getEventType` metode.

```
System.out.println("Executing the decision task for the history events: [");
for (HistoryEvent event : events) {
    System.out.println(" " + event);
    switch(event.getEventType()) {
        case "WorkflowExecutionStarted":
            workflow_input =
                event.getWorkflowExecutionStartedEventAttributes()
                    .getInput();
            break;
        case "ActivityTaskScheduled":
            scheduled_activities++;
            break;
        case "ScheduleActivityTaskFailed":
            scheduled_activities--;
            break;
        case "ActivityTaskStarted":
            scheduled_activities--;
    }
}
```

```
        open_activities++;
        break;
    case "ActivityTaskCompleted":
        open_activities--;
        activity_completed = true;
        result = event.getActivityTaskCompletedEventAttributes()
                .getResult();
        break;
    case "ActivityTaskFailed":
        open_activities--;
        break;
    case "ActivityTaskTimedOut":
        open_activities--;
        break;
    }
}
System.out.println("]");
```

Untuk keperluan alur kerja kami, kami paling tertarik pada:

- acara "WorkflowExecutionStarted", yang menunjukkan bahwa eksekusi alur kerja telah dimulai (biasanya berarti bahwa Anda harus menjalankan aktivitas pertama dalam alur kerja), dan yang menyediakan input awal yang disediakan untuk alur kerja. Dalam kasus ini, ini adalah bagian nama salam kita, jadi disimpan dalam String untuk digunakan saat menjadwalkan aktivitas yang akan dijalankan.
- acara "ActivityTaskCompleted", yang dikirim setelah aktivitas yang dijadwalkan selesai. Data peristiwa juga mencakup nilai pengembalian aktivitas yang telah selesai. Karena kita hanya memiliki satu aktivitas, kita akan menggunakan nilai itu sebagai hasil dari seluruh alur kerja.

Jenis acara lainnya dapat digunakan jika alur kerja Anda memerlukannya. Lihat deskripsi [HistoryEvent](#) kelas untuk informasi tentang setiap jenis acara.

+ CATATAN: String dalam switch pernyataan diperkenalkan di Jawa 7. Jika Anda menggunakan versi Java sebelumnya, Anda dapat menggunakan [EventType](#) kelas untuk mengonversi yang String dikembalikan `history_event.getType()` ke nilai enum dan kemudian kembali ke String jika perlu:

```
EventType et = EventType.fromValue(event.getEventType());
```

1. Setelah switch pernyataan, tambahkan lebih banyak kode untuk merespons dengan keputusan yang tepat berdasarkan tugas yang diterima.

```

if (activity_completed) {
    decisions.add(
        new Decision()
            .withDecisionType(DecisionType.CompleteWorkflowExecution)
            .withCompleteWorkflowExecutionDecisionAttributes(
                new CompleteWorkflowExecutionDecisionAttributes()
                    .withResult(result)));
} else {
    if (open_activities == 0 && scheduled_activities == 0) {

        ScheduleActivityTaskDecisionAttributes attrs =
            new ScheduleActivityTaskDecisionAttributes()
                .withActivityType(new ActivityType()
                    .withName>HelloTypes.ACTIVITY)
                    .withVersion>HelloTypes.ACTIVITY_VERSION))
                .withActivityId(UUID.randomUUID().toString())
                .withInput(workflow_input);

        decisions.add(
            new Decision()
                .withDecisionType(DecisionType.ScheduleActivityTask)
                .withScheduleActivityTaskDecisionAttributes(attrs));
    } else {
        // an instance of HelloActivity is already scheduled or running. Do nothing,
        another
        // task will be scheduled once the activity completes, fails or times out
    }
}

System.out.println("Exiting the decision task with the decisions " + decisions);

```

- Jika aktivitas belum dijadwalkan, kami merespons dengan `ScheduleActivityTask` keputusan, yang memberikan informasi dalam [ScheduleActivityTaskDecisionAttributes](#) struktur tentang aktivitas yang Amazon SWF harus dijadwalkan berikutnya, juga termasuk data apa pun yang Amazon SWF harus dikirim ke aktivitas.
- Jika aktivitas selesai, maka kami mempertimbangkan seluruh alur kerja selesai dan merespons dengan `CompletedWorkflowExecution` keputusan, mengisi

[CompleteWorkflowExecutionDecisionAttributes](#) struktur untuk memberikan rincian tentang alur kerja yang telah selesai. Dalam kasus ini, kami mengembalikan hasil aktivitas.

Dalam kedua kasus, informasi keputusan ditambahkan ke `Decision` daftar yang dinyatakan di bagian atas metode.

2. Selesaikan tugas keputusan dengan mengembalikan daftar `Decision` objek yang dikumpulkan saat memproses tugas. Tambahkan kode ini di akhir `executeDecisionTask` metode yang telah kita tulis:

```
swf.respondDecisionTaskCompleted(  
    new RespondDecisionTaskCompletedRequest()  
        .withTaskToken(taskToken)  
        .withDecisions(decisions));
```

`respondDecisionTaskCompleted` metode SWF klien mengambil token tugas yang mengidentifikasi tugas serta daftar objek `Decision`.

Menerapkan starter alur kerja

Akhirnya, kita akan menulis beberapa kode untuk memulai eksekusi alur kerja.

1. Buka editor teks Anda dan buat `fileWorkflowStarter.java`, tambahkan deklarasi paket dan impor sesuai dengan langkah-langkah [umum](#).
2. Tambahkan `WorkflowStarter` kelas:

```
package aws.example.helloswf;  
  
import com.amazonaws.regions.Regions;  
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;  
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;  
import com.amazonaws.services.simpleworkflow.model.*;  
  
public class WorkflowStarter {  
    private static final AmazonSimpleWorkflow swf =  
  
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();  
    public static final String WORKFLOW_EXECUTION = "HelloWorldWorkflowExecution";  
  
    public static void main(String[] args) {
```

```

String workflow_input = "{SWF}";
if (args.length > 0) {
    workflow_input = args[0];
}

System.out.println("Starting the workflow execution '" + WORKFLOW_EXECUTION +
    "' with input '" + workflow_input + "'.");

WorkflowType wf_type = new WorkflowType()
    .withName(HelloTypes.WORKFLOW)
    .withVersion(HelloTypes.WORKFLOW_VERSION);

Run run = swf.startWorkflowExecution(new StartWorkflowExecutionRequest()
    .withDomain(HelloTypes.DOMAIN)
    .withWorkflowType(wf_type)
    .withWorkflowId(WORKFLOW_EXECUTION)
    .withInput(workflow_input)
    .withExecutionStartToCloseTimeout("90"));

System.out.println("Workflow execution started with the run id '" +
    run.getRunId() + "'.");
}
}

```

`WorkflowStarterKelas` terdiri dari metode tunggal, `main`, yang mengambil argumen opsional diteruskan pada baris perintah sebagai data input untuk alur kerja.

Metode klien SWF, `startWorkflowExecution`, mengambil [StartWorkflowExecutionRequest](#) objek sebagai masukan. Di sini, selain menentukan jenis domain dan alur kerja yang akan dijalankan, kami menyediakannya dengan:

- nama eksekusi alur kerja yang dapat dibaca manusia
- alur kerja input data (disediakan pada baris perintah dalam contoh kita)
- nilai batas waktu yang mewakili berapa lama, dalam hitungan detik, bahwa seluruh alur kerja harus mengambil untuk menjalankan.

Objek [Run](#) yang `startWorkflowExecution` mengembalikan menyediakan ID run, nilai yang dapat digunakan untuk mengidentifikasi eksekusi alur kerja tertentu ini dalam Amazon SWF sejarah eksekusi alur kerja Anda.

+ CATATAN: ID run dihasilkan oleh Amazon SWF, dan tidak sama dengan nama eksekusi alur kerja yang Anda lulus ketika memulai eksekusi alur kerja.

Membangun contoh

Untuk membangun proyek contoh dengan Maven, pergi ke `helloswf` direktori dan ketik:

```
mvn package
```

Hasilnya `helloswf-1.0.jar` akan dihasilkan dalam `target` direktori.

Jalankan contoh

Contoh ini terdiri dari empat kelas executable terpisah, yang dijalankan secara independen satu sama lain.

Note

Jika Anda menggunakan sistem Linux, macOS, atau Unix, Anda dapat menjalankan semuanya, satu demi satu, dalam satu jendela terminal. Jika Anda menjalankan Windows, Anda harus membuka dua contoh baris perintah tambahan dan menavigasi ke `helloswf` direktori di masing-masing.

Mengatur classpath Java

Meskipun Maven telah menangani dependensi untuk Anda, untuk menjalankan contoh, Anda harus menyediakan perpustakaan AWS SDK dan dependensinya pada classpath Java Anda. Anda dapat mengatur variabel `CLASSPATH` lingkungan ke lokasi pustaka AWS SDK dan `third-party/lib` direktori di SDK, yang mencakup dependensi yang diperlukan:

```
export CLASSPATH='target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/*'  
java example.swf.hello.HelloTypes
```

atau gunakan `-cp` opsi `java` perintah untuk mengatur classpath saat menjalankan setiap aplikasi.

```
java -cp target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/* \  
example.swf.hello.HelloTypes
```

Gaya yang Anda gunakan terserah Anda. Jika Anda tidak kesulitan membangun kode, keduanya kemudian mencoba untuk menjalankan contoh dan mendapatkan serangkaian "NoClassDefFound" kesalahan, kemungkinan karena classpath diatur secara tidak benar.

Mendaftarkan jenis domain, alur kerja, dan jenis aktivitas

Sebelum menjalankan pekerja dan starter alur kerja, Anda harus mendaftarkan domain dan alur kerja serta jenis aktivitas Anda. Kode untuk melakukan ini diimplementasikan dalam [Mendaftarkan alur kerja domain dan jenis aktivitas](#).

Setelah membangun, dan jika Anda telah [mengatur CLASSPATH](#), Anda dapat menjalankan kode registrasi dengan menjalankan perintah:

```
echo 'Supply the name of one of the example classes as an argument.'
```

Memulai aktivitas dan alur kerja

Setelah jenis didaftarkan, Anda dapat memulai aktivitas dan alur kerja pekerja. Ini akan terus berjalan dan polling untuk tugas sampai mereka mati, jadi Anda harus menjalankannya di jendela terminal terpisah, atau, jika Anda berjalan di Linux, macOS, atau Unix Anda dapat menggunakan & operator untuk menyebabkan masing-masing dari mereka untuk menelurkan proses terpisah ketika dijalankan.

```
echo 'If there are arguments to the class, put them in quotes after the class  
name.'  
exit 1
```

Jika Anda menjalankan perintah ini di jendela terpisah, hilangkan & operator akhir dari setiap baris.

Memulai eksekusi alur kerja

Sekarang setelah pekerja aktivitas dan alur kerja Anda melakukan polling, Anda dapat memulai eksekusi alur kerja. Proses ini akan berjalan sampai alur kerja mengembalikan status selesai. Anda harus menjalankannya di jendela terminal baru (kecuali jika Anda menjalankan pekerja Anda sebagai proses melahirkan baru dengan menggunakan & operator).

```
fi
```

Note

Jika Anda ingin memberikan data input Anda sendiri, yang akan diteruskan terlebih dahulu ke alur kerja dan kemudian ke aktivitas, tambahkan ke baris perintah. Misalnya:

```
echo "## Running $className..."
```

Setelah Anda memulai eksekusi alur kerja, Anda harus mulai melihat output yang disampaikan oleh kedua pekerja dan oleh eksekusi alur kerja itu sendiri. Ketika alur kerja akhirnya selesai, outputnya akan dicetak ke layar.

Sumber lengkap untuk contoh ini

Anda dapat menelusuri [sumber lengkap](#) untuk contoh ini di Github di repositori. `aws-java-developer-guide`

Untuk informasi selengkapnya

- Para pekerja yang disajikan di sini dapat mengakibatkan kehilangan tugas jika mereka shutdown sementara jajak pendapat alur kerja masih berlangsung. Untuk mengetahui cara mematikan pekerja dengan anggun, lihat Menutup Pekerja [Aktivitas dan Alur Kerja dengan Anggun](#).
- Untuk mempelajari lebih lanjut Amazon SWF, kunjungi [Amazon SWF](#) halaman beranda atau lihat [Panduan Amazon SWF Pengembang](#).
- Anda dapat menggunakan AWS Flow Framework for Java untuk menulis alur kerja yang lebih kompleks dalam gaya Java yang elegan menggunakan anotasi. Untuk mempelajari lebih lanjut, lihat [Panduan Pengembang AWS Flow Framework untuk Java](#).

Tugas Lambda

Sebagai alternatif untuk, atau dalam hubungannya dengan, Amazon SWF kegiatan, Anda dapat menggunakan [Lambda](#) berfungsi untuk mewakili unit kerja dalam alur kerja Anda, dan menjadwalkan mereka sama dengan kegiatan.

Topik ini berfokus pada bagaimana menerapkan Amazon SWF Lambda tugas menggunakan AWS SDK for Java. Untuk informasi lebih lanjut tentang Lambda tugas secara umum, lihat [AWS Lambda Tugas](#) di Amazon SWF Panduan Developer.

Menyiapkan IAM role untuk menjalankan fungsi Lambda

Sebelum Amazon SWF dapat menjalankan Lambda fungsi, Anda perlu mengatur peran IAM untuk memberikan Amazon SWF izin untuk menjalankan Lambda fungsi atas nama Anda. Untuk informasi lengkap tentang cara melakukan ini, lihat [AWS Lambda Tugas](#).

Anda memerlukan Amazon Resource Name (ARN) dari IAM role ini ketika mendaftarkan alur kerja yang akan digunakan Lambda tugas.

BuatLambdafungsi

Anda dapat menulisLambdafungsi dalam sejumlah bahasa yang berbeda, termasuk Java. Untuk informasi lengkap tentang cara menulis, menyebarkan, dan menggunakanLambdafungsi, lihat[AWS LambdaPanduan Pengembang](#).

Note

Tidak peduli bahasa apa yang Anda gunakan untuk menulisLambdafungsi, dapat dijadwalkan dan dijalankan olehapa pun Amazon SWFalur kerja, terlepas dari bahasa yang ditulis kode alur kerja Anda.Amazon SWFmenangani rincian menjalankan fungsi dan meneruskan data ke dan dari itu.

Berikut adalah sederhanaLambdafungsi yang dapat digunakan di tempat kegiatan di[Membangun SederhanaAmazon SWFAplikasi](#).

- Versi ini ditulis dalam JavaScript, yang dapat dimasukkan langsung menggunakan[AWS Management Console](#):

```
exports.handler = function(event, context) {
    context.succeed("Hello, " + event.who + "!");
};
```

- Berikut adalah fungsi yang sama ditulis di Java, yang juga bisa Anda gunakan dan jalankan di Lambda:

```
package example.swf.hellolambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.util.json.JSONException;
import com.amazonaws.util.json.JSONObject;

public class SwfHelloLambdaFunction implements RequestHandler<Object, Object> {
    @Override
    public Object handleRequest(Object input, Context context) {
        String who = "{SWF}";
        if (input != null) {
            JSONObject jso = null;
            try {
```

```

        jso = new JSONObject(input.toString());
        who = jso.getString("who");
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
return ("Hello, " + who + "!");
}
}

```

Note

Untuk mempelajari lebih lanjut tentang menerapkan fungsi Java ke Lambda, lihat [Membuat Paket Deployment \(Java\)](#) di AWS Lambda Panduan Developer. Anda juga akan ingin melihat bagian berjudul [Model Pemrograman untuk Authoring Lambda Fungsi di Java](#).

Lambda mengambil peristiwa atau memasukkan objek sebagai parameter pertama, dan konteks objek sebagai yang kedua, yang memberikan informasi tentang permintaan untuk menjalankan Lambda fungsi. Fungsi khusus ini mengharapkan masukan untuk berada di JSON, dengan `who` bidang diatur ke nama yang digunakan untuk membuat ucapan.

Daftarkan alur kerja untuk digunakan dengan Lambda

Untuk alur kerja untuk menjadwalkan Lambda fungsi, Anda harus memberikan nama peran IAM yang menyediakan Amazon SWF dengan izin untuk memohon Lambda fungsi. Anda dapat mengatur ini selama pendaftaran alur kerja dengan menggunakan `withDefaultLambdaRole` atau `setDefaultLambdaRole` metode [RegisterWorkflowTypeRequest](#).

```

System.out.println("*** Registering the workflow type '" + WORKFLOW + "' - " +
    WORKFLOW_VERSION
    + "'.");
try {
    swf.registerWorkflowType(new RegisterWorkflowTypeRequest()
        .withDomain(DOMAIN)
        .withName(WORKFLOW)
        .withDefaultLambdaRole(lambda_role_arn)
        .withVersion(WORKFLOW_VERSION)
        .withDefaultChildPolicy(ChildPolicy.TERMINATE)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskStartToCloseTimeout("30"));
}

```

```

}
catch (TypeAlreadyExistsException e) {

```

MenjadwalkanLambdatugas

MenjadwalkanLambdatugas mirip dengan penjadwalan aktivitas. Anda memberikan [Keputusan](#) dengan `ScheduleLambdaFunction`DecisionType` dan dengan [ScheduleLambdaFunctionDecisionAttributes](#).

```

running_functions == 0 && scheduled_functions == 0) {
AWSLambda lam = AWSLambdaClientBuilder.defaultClient();
GetFunctionConfigurationResult function_config =
    lam.getFunctionConfiguration(
        new GetFunctionConfigurationRequest()
            .withFunctionName("HelloFunction"));
String function_arn = function_config.getFunctionArn();

ScheduleLambdaFunctionDecisionAttributes attrs =
    new ScheduleLambdaFunctionDecisionAttributes()
        .withId("HelloFunction (Lambda task example)")
        .withName(function_arn)
        .withInput(workflow_input);

decisions.add(

```

Di `ScheduleLambdaFunctionDecisionAttributes`, Anda harus menyediakan nama, yang merupakan ARN dari Lambda fungsi untuk memanggil, dan id, yang merupakan nama yang Amazon SWF akan digunakan untuk mengidentifikasi Lambda fungsi dalam log sejarah.

Anda juga dapat memberikan opsional memasukkan untuk Lambda fungsi dan mengatur nyamulai menutup batas waktu nilai, yang merupakan jumlah detik yang Lambda fungsi diperbolehkan untuk menjalankan sebelum menghasilkan `LambdaFunctionTimedOut` peristiwa.

Note

Kode ini menggunakan [AWSLambdaClient](#) untuk mengambil ARN dari Lambda fungsi, mengingat nama fungsi. Anda dapat menggunakan teknik ini untuk menghindari hard-coding ARN penuh (yang mencakup Anda Akun AWS ID) dalam kode Anda.

Tangani event fungsi Lambda di decider Anda

Lambdatugas akan menghasilkan sejumlah peristiwa yang dapat Anda ambil tindakan saat melakukan pemungutan suara untuk tugas keputusan di pekerja alur kerja Anda, sesuai dengan siklus hidup AndaLambdatugas, dengan [EventTypenilai-nilai](#) seperti `LambdaFunctionScheduled`, `LambdaFunctionStarted`, dan `LambdaFunctionCompleted`. Jika `Lambda` fungsi gagal, atau membutuhkan waktu lebih lama untuk menjalankan dari nilai batas waktu yang ditetapkan, Anda akan menerima baik `LambdaFunctionFailed` atau `LambdaFunctionTimedOut` jenis acara, masing-masing.

```
boolean function_completed = false;
String result = null;

System.out.println("Executing the decision task for the history events: [");
for (HistoryEvent event : events) {
    System.out.println("  " + event);
    EventType event_type = EventType.fromValue(event.getEventType());
    switch(event_type) {
    case WorkflowExecutionStarted:
        workflow_input =
            event.getWorkflowExecutionStartedEventAttributes()
                .getInput();
        break;
    case LambdaFunctionScheduled:
        scheduled_functions++;
        break;
    case ScheduleLambdaFunctionFailed:
        scheduled_functions--;
        break;
    case LambdaFunctionStarted:
        scheduled_functions--;
        running_functions++;
        break;
    case LambdaFunctionCompleted:
        running_functions--;
        function_completed = true;
        result = event.getLambdaFunctionCompletedEventAttributes()
            .getResult();
        break;
    case LambdaFunctionFailed:
        running_functions--;
        break;
    }
```

```
case LambdaFunctionTimedOut:
    running_functions--;
    break;
```

Menerima output dari Lambda fungsi

Ketika Anda menerima `LambdaFunctionCompleted` [`EventType`](#), you can retrieve your `0` function's return value by first calling `getLambdaFunctionCompletedEventAttributes` pada [`HistoryEvent`](#) untuk mendapatkan [`LambdaFunctionCompletedEventAttributes`](#) objek, dan kemudian memanggil `getResult` metode untuk mengambil output dari Lambda Fungsi:

```
LambdaFunctionCompleted:
running_functions--;
```

Sumber lengkap untuk contoh ini

Anda dapat menelusuri sumber lengkap: `github: <awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf_lambda/>` untuk contoh ini di Github di `aws-java-pengembang-panduan` repositori.

Mematikan Kegiatan dan Alur Kerja Pekerja Anggun

Parameter [`Membangun Sederhana Amazon SWF Aplikasi`](#) topik yang disediakan implementasi lengkap dari aplikasi alur kerja sederhana yang terdiri dari aplikasi pendaftaran, aktivitas dan alur kerja pekerja, dan alur kerja starter.

Kelas pekerja dirancang untuk berjalan terus menerus, polling untuk tugas yang dikirim oleh Amazon SWF untuk menjalankan kegiatan atau keputusan kembali. Setelah permintaan jajak pendapat dibuat, Amazon SWF mencatat poller dan akan mencoba untuk menetapkan tugas untuk itu.

Jika pekerja alur kerja dihentikan selama polling panjang, Amazon SWF mungkin masih mencoba mengirim tugas ke pekerja yang diakhiri, menghasilkan tugas yang hilang (sampai tugas habis).

Salah satu cara untuk menangani situasi ini adalah menunggu semua permintaan jajak pendapat panjang untuk kembali sebelum pekerja berakhir.

Dalam topik ini, kami akan menulis ulang `activity worker` dari `helloswf`, menggunakan kait `shutdown` Java untuk mencoba `shutdown` yang anggun dari pekerja aktivitas.

Berikut adalah kode lengkapnya:

```
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.ActivityTask;
import com.amazonaws.services.simpleworkflow.model.PollForActivityTaskRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskCompletedRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskFailedRequest;
import com.amazonaws.services.simpleworkflow.model.TaskList;

public class ActivityWorkerWithGracefulShutdown {

    private static final AmazonSimpleWorkflow swf =

AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
    private static final CountDownLatch waitForTermination = new CountDownLatch(1);
    private static volatile boolean terminate = false;

    private static String executeActivityTask(String input) throws Throwable {
        return "Hello, " + input + "!";
    }

    public static void main(String[] args) {
        Runtime.getRuntime().addShutdownHook(new Thread() {
            @Override
            public void run() {
                try {
                    terminate = true;
                    System.out.println("Waiting for the current poll request" +
                        " to return before shutting down.");
                    waitForTermination.await(60, TimeUnit.SECONDS);
                }
                catch (InterruptedException e) {
                    // ignore
                }
            }
        });
        try {
            pollAndExecute();
        }
    }
}
```

```
        finally {
            waitForTermination.countDown();
        }
    }

    public static void pollAndExecute() {
        while (!terminate) {
            System.out.println("Polling for an activity task from the tasklist '"
                + HelloTypes.TASKLIST + "' in the domain '"
                + HelloTypes.DOMAIN + "'.");

            ActivityTask task = swf.pollForActivityTask(new
                PollForActivityTaskRequest()
                    .withDomain(HelloTypes.DOMAIN)
                    .withTaskList(new TaskList().withName(HelloTypes.TASKLIST)));

            String taskToken = task.getTaskToken();

            if (taskToken != null) {
                String result = null;
                Throwable error = null;

                try {
                    System.out.println("Executing the activity task with input '"
                        + task.getInput() + "'.");
                    result = executeActivityTask(task.getInput());
                }
                catch (Throwable th) {
                    error = th;
                }

                if (error == null) {
                    System.out.println("The activity task succeeded with result '"
                        + result + "'.");
                    swf.respondActivityTaskCompleted(
                        new RespondActivityTaskCompletedRequest()
                            .withTaskToken(taskToken)
                            .withResult(result));
                }
                else {
                    System.out.println("The activity task failed with the error '"
                        + error.getClass().getSimpleName() + "'.");
                    swf.respondActivityTaskFailed(
                        new RespondActivityTaskFailedRequest()

```



```
}  
}
```

Daftar domain

Anda dapat daftar [Amazon SWF](#) domain yang terkait dengan akun Anda dan AWS wilayah berdasarkan jenis registrasi.

1. Buat [ListDomainsRequest](#) objek, dan tentukan status pendaftaran domain yang Anda minati—ini diperlukan.
2. PANGLAN [AmazonSimpleWorkflowClient.ListDomains](#) dengan [ListDomainRequest](#) objek. Hasil disediakan dalam [DomainInfos](#) objek.
3. PANGLAN [GetDomainInfos](#) pada objek kembali untuk mendapatkan daftar [DomainInfo](#) objek.
4. PANGLAN [getName](#) pada masing-masing [DomainInfo](#) keberatan untuk mendapatkan namanya.

Kode berikut menunjukkan prosedur ini:

```
public void list_swf_domains(AmazonSimpleWorkflowClient swf)
{
    ListDomainsRequest request = new ListDomainsRequest();
    request.setRegistrationStatus("REGISTERED");
    DomainInfos domains = swf.listDomains(request);
    System.out.println("Current Domains:");
    for (DomainInfo di : domains.getDomainInfos())
    {
        System.out.println(" * " + di.getName());
    }
}
```

Contoh Kode disertakan dengan SDK

Parameter AWS SDK for Java hadir dikemas dengan sampel kode yang menunjukkan banyak fitur SDK dalam program buildable dan runnable. Anda dapat mempelajari atau memodifikasi ini untuk mengimplementasikan Anda sendiri AWS solusi menggunakan AWS SDK for Java.

Cara Mendapatkan Sampel

Parameter AWS SDK for Java sampel kode disediakan dalam direktori SDK. Jika Anda mengunduh dan menginstal SDK menggunakan informasi di [Menyiapkan AWS SDK for Java](#), Anda sudah memiliki sampel pada sistem Anda.

Anda juga dapat melihat sampel terbaru di AWS SDK for Java Repositori GitHub, di [src/sampel](#) direktori.

Membangun dan Menjalankan Sampel Menggunakan Command Line

Sampel termasuk [Semut](#) membangun skrip sehingga Anda dapat dengan mudah membangun dan menjalankannya dari baris perintah. Setiap sampel juga berisi file README dalam format HTML yang berisi informasi khusus untuk setiap sampel.

Note

Jika Anda menelusuri kode contoh di GitHub, klik [Mentah](#) tombol di tampilan kode sumber saat melihat file README.html sampel. Dalam mode mentah, HTML akan membuat seperti yang dimaksudkan di browser Anda.

Prasyarat

Sebelum menjalankan salah satu AWS SDK for Java sampel, Anda perlu mengatur AWS kredensi di lingkungan atau dengan AWS CLI, seperti yang ditentukan dalam [Mengatur AWS Kredensial dan Wilayah untuk Pembangunan](#). Sampel menggunakan rantai penyedia kredensi default bila memungkinkan. Jadi dengan menetapkan kredensi Anda dengan cara ini, Anda dapat menghindari praktik berisiko memasukkan AWS kredensi dalam file dalam direktori kode sumber (di mana mereka secara tidak sengaja dapat diperiksa dan dibagikan secara tidak sengaja).

Menjalankan Sampel

1. Ubah ke direktori yang berisi kode sampel. Misalnya, jika Anda berada di direktori root AWS SDK download dan ingin menjalankan `AwsConsoleApp` sampel, Anda akan mengetik:

```
cd samples/AwsConsoleApp
```

2. Bangun dan jalankan sampel dengan Ant. Target build default melakukan kedua tindakan, sehingga Anda bisa memasukkan:

```
ant
```

Sampel mencetak informasi ke output standar—misalnya:

```
=====
Welcome to the {AWS} Java SDK!
=====
You have access to 4 Availability Zones.

You have 0 {EC2} instance(s) running.

You have 13 Amazon SimpleDB domain(s) containing a total of 62 items.

You have 23 {S3} bucket(s), containing 44 objects with a total size of 154767691 bytes.
```

Membangun dan Menjalankan Sampel Menggunakan Eclipse IDE

Jika Anda menggunakan AWS Toolkit for Eclipse, Anda juga dapat memulai proyek baru di Eclipse berdasarkan AWS SDK for Java atau tambahkan SDK ke proyek Java yang ada.

Prasyarat

Setelah menginstal AWS Toolkit for Eclipse, kami sarankan untuk mengonfigurasi Toolkit dengan kredensi keamanan Anda. Anda dapat melakukannya kapan saja dengan memilih Preferensi dari Jendela menu di Eclipse, dan kemudian memilih AWS Toolkit bagian.

Menjalankan Sampel

1. Buka Eclipse.
2. Membuat AWS Proyek Java. Dalam Eclipse, di Berkas menu, pilih Baru, dan kemudian klik Proyek. Parameter Proyek Baru Penyihir terbuka.
3. Perluas AWS kategori, lalu pilih AWS Proyek Java.
4. Pilih Selanjutnya. Halaman pengaturan proyek ditampilkan.
5. Masukkan nama di Nama Proyek kotak. Parameter AWS SDK for Java Kelompok sampel menampilkan sampel yang tersedia di SDK, seperti yang dijelaskan sebelumnya.

6. Pilih sampel yang ingin Anda sertakan dalam proyek Anda dengan memilih setiap kotak centang.
7. Masukkan AWS kredensial. Jika Anda sudah mengkonfigurasi AWS Toolkit for Eclipse dengan kredensial Anda, ini secara otomatis diisi.
8. Pilih Selesai. Proyek ini dibuat dan ditambahkan ke Explorer Proyek.
9. Pilih sampel .java file yang ingin Anda jalankan. Misalnya, untuk Amazon S3 sampel, pilih `S3Sample.java`.
10. Pilih Jalankan dari Jalankan menu.
11. Klik kanan proyek di Explorer Proyek, arahkan ke Jalur Bangun, dan kemudian pilih Tambahkan Pustaka.
12. Pilih AWSSDK Java, pilih Selanjutnya, lalu ikuti petunjuk di layar yang tersisa.

Keamanan untuk AWS SDK for Java

Keamanan cloud di Amazon Web Services (AWS) merupakan prioritas tertinggi. Sebagai seorang pelanggan AWS, Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan dari organisasi yang paling sensitif terhadap keamanan. Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model Tanggung Jawab Bersama](#) menggambarkan ini sebagai Keamanan dari Cloud dan Keamanan dalam Cloud.

Security of the Cloud - AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan semua layanan yang ditawarkan di AWS Cloud dan memberi Anda layanan yang dapat Anda gunakan dengan aman. Tanggung jawab keamanan kami adalah prioritas tertinggi di AWS, dan efektivitas keamanan kami secara teratur diuji dan diverifikasi oleh auditor pihak ketiga sebagai bagian dari [Program AWS Kepatuhan](#).

Keamanan di Cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan, dan faktor-faktor lain termasuk sensitivitas data Anda, persyaratan organisasi Anda, serta undang-undang dan peraturan yang berlaku.

AWS Produk atau layanan ini mengikuti [model tanggung jawab bersama](#) melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat [halaman dokumentasi keamanan AWS layanan](#) dan [AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan](#).

Topik

- [Perlindungan data di AWS SDK for Java 1.x](#)
- [AWS SDK for Java dukungan untuk TLS](#)
- [Identity and Access Management](#)
- [Validasi Kepatuhan untuk AWS Produk atau Layanan ini](#)
- [Ketahanan untuk AWS Produk atau Layanan ini](#)
- [Keamanan Infrastruktur untuk AWS Produk atau Layanan ini](#)
- [Amazon S3 Migrasi Klien Enkripsi](#)

Perlindungan data di AWS SDK for Java 1.x

[Model tanggung jawab bersama](#) berlaku untuk perlindungan data dalam AWS produk atau layanan ini. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur

global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali terhadap konten yang di-hosting pada infrastruktur ini. Konten ini meliputi konfigurasi keamanan dan tugas-tugas manajemen untuk berbagai layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat [FAQ Privasi Data](#). Untuk informasi tentang perlindungan data di Eropa, lihat [Model Tanggung Jawab AWS Bersama dan posting blog GDPR](#) di Blog AWS Keamanan.

Untuk tujuan perlindungan data, kami menyarankan Anda untuk melindungi Akun AWS kredensial dan menyiapkan akun pengguna individual dengan AWS Identity and Access Management (IAM). Dengan cara ini, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugas mereka. Kami juga merekomendasikan agar Anda mengamankan data Anda dengan cara-cara berikut ini:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, dengan semua kontrol keamanan default dalam AWS layanan.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan. Amazon S3
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi lebih lanjut tentang titik akhir FIPS yang tersedia, lihat [Standar Pemrosesan Informasi Federal \(FIPS\) 140-2](#).

Kami sangat merekomendasikan agar Anda tidak memasukkan informasi identifikasi sensitif apapun, seperti nomor rekening pelanggan Anda, ke dalam kolom isian teks bebas seperti kolom Nama. Ini termasuk saat Anda bekerja dengan AWS produk atau layanan ini atau AWS layanan lain menggunakan konsol, API AWS CLI, atau AWS SDK. Setiap data yang Anda masukkan ke dalam AWS produk atau layanan ini atau layanan lain mungkin diambil untuk dimasukkan dalam log diagnostik. Saat Anda memberikan URL ke server eksternal, jangan sertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

AWS SDK for Java dukungan untuk TLS

Informasi berikut hanya berlaku untuk implementasi Java SSL (implementasi SSL default di AWS SDK for Java). Jika Anda menggunakan implementasi SSL yang berbeda, lihat implementasi SSL spesifik Anda untuk mempelajari cara menerapkan versi TLS.

Cara memeriksa versi TLS

Konsultasikan dokumentasi penyedia mesin virtual Java (JVM) Anda untuk menentukan versi TLS mana yang didukung di platform Anda. Untuk beberapa JVM, kode berikut akan mencetak versi SSL mana yang didukung.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

Untuk melihat jabat tangan SSL beraksi dan versi TLS apa yang digunakan, Anda dapat menggunakan properti sistem `javax.net.debug`.

```
java app.jar -Djavax.net.debug=ssl
```

Note

TLS 1.3 tidak kompatibel dengan SDK for Java versi 1.9.5 hingga 1.10.31. Untuk informasi lebih lanjut, lihat posting blog berikut.

<https://aws.amazon.com/blogs/developer/tls-1-3--incompatibility-with-aws-sdk-for-java-versions-1-9-5-ke-1-10-31/>

Menetapkan versi TLS minimum

SDK selalu lebih menyukai versi TLS terbaru yang didukung oleh platform dan layanan. Jika Anda ingin menerapkan versi TLS minimum tertentu, lihat dokumentasi JVM Anda. Untuk JVM berbasis OpenJDK, Anda dapat menggunakan properti sistem `jdk.tls.client.protocols`

```
java app.jar -Djdk.tls.client.protocols=PROTOCOLS
```

Konsultasikan dokumentasi JVM Anda untuk mengetahui nilai PROTOKOL yang didukung.

Identity and Access Management

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. AWS IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Bagaimana Layanan AWS bekerja dengan IAM](#)
- [Memecahkan masalah AWS identitas dan akses](#)

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan. AWS

Pengguna layanan — Jika Anda menggunakan Layanan AWS untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensial dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak AWS fitur untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur AWS, lihat [Memecahkan masalah AWS identitas dan akses](#) atau panduan pengguna yang Layanan AWS Anda gunakan.

Administrator layanan — Jika Anda bertanggung jawab atas AWS sumber daya di perusahaan Anda, Anda mungkin memiliki akses penuh ke AWS. Tugas Anda adalah menentukan AWS fitur dan sumber daya mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep 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 terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat [Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS](#).

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. 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 selengkapnya, lihat [Autentikasi multi-faktor](#) dalam Panduan Pengguna AWS IAM Identity Center dan [Menggunakan autentikasi multi-faktor \(MFA\) dalam 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. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensial sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensial yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat [Apakah itu Pusat Identitas IAM?](#) dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat [Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) dalam Panduan Pengguna IAM.

[Grup IAM](#) adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat [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 cara menggunakan peran, lihat [Menggunakan peran IAM](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Membuat peran untuk Penyedia Identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat [Set izin](#) dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya](#) dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
 - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan

AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).

- Peran layanan – Peran layanan adalah [peran IAM](#) yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS](#) dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola 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 dalam 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 [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat 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 yang dikelola atau kebijakan inline, lihat [Memilih antara kebijakan yang dikelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus

[menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat [Gambaran umum daftar kontrol akses \(ACL\)](#) dalam Panduan Developer Amazon Simple Storage Service.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- **Batasan izin** – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- **Kebijakan kontrol layanan (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 di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat [Cara kerja SCP](#) dalam Panduan Pengguna AWS Organizations .
- **Kebijakan sesi** – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi.

Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

Bagaimana Layanan AWS bekerja dengan IAM

Untuk mendapatkan tampilan tingkat tinggi tentang cara Layanan AWS 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 orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah AWS mendukung fitur-fitur ini, lihat [Bagaimana Layanan AWS bekerja dengan IAM](#).
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat [Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat [Menyediakan akses ke pengguna terautentikasi eksternal \(federasi identitas\)](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan 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.

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 sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat [Referensi kontrol Security Hub](#).
- [Amazon GuardDuty](#) — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas yang mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- [AWS Audit Manager](#)Ini Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

AWS Produk atau layanan ini mengikuti [model tanggung jawab bersama](#) melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat [halaman dokumentasi keamanan AWS layanan](#) dan [AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan](#).

Ketahanan untuk AWS Produk atau Layanan ini

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones.

Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan.

Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat [Infrastruktur AWS Global](#).

AWS Produk atau layanan ini mengikuti [model tanggung jawab bersama](#) melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat [halaman dokumentasi keamanan AWS layanan](#) dan [AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan](#).

Keamanan Infrastruktur untuk AWS Produk atau Layanan ini

AWS Produk atau layanan ini menggunakan layanan terkelola, dan karenanya dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat [Keamanan AWS Cloud](#). Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat [Perlindungan Infrastruktur dalam Kerangka Kerja](#) yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS Produk atau Layanan ini melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan [AWS Security Token](#)

[Service](#) (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

AWS Produk atau layanan ini mengikuti [model tanggung jawab bersama](#) melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat [halaman dokumentasi keamanan AWS layanan](#) dan [AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan](#).

Amazon S3 Migrasi Klien Enkripsi

Topik ini menunjukkan cara memigrasikan aplikasi Anda dari Versi 1 (V1) klien enkripsi Amazon Simple Storage Service (Amazon S3) ke Versi 2 (V2) dan memastikan ketersediaan aplikasi selama proses migrasi.

Prasyarat

Amazon S3 enkripsi sisi klien memerlukan yang berikut:

- Java 8 atau yang lebih baru diinstal di lingkungan aplikasi Anda. [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 JDK. AdoptOpen](#)
- Paket [Bouncy Castle Crypto](#). Anda dapat menempatkan file Bouncy Castle .jar di classpath lingkungan aplikasi Anda, atau menambahkan dependensi pada ArtifactID (dengan groupId dari ke file Maven `bcprov-ext-jdk15on` Anda. `org.bouncycastle pom.xml`

Ikhtisar Migrasi

Migrasi ini terjadi dalam dua fase:

1. Perbarui klien yang ada untuk membaca format baru. Perbarui aplikasi Anda untuk menggunakan versi 1.11.837 atau yang lebih baru AWS SDK for Java dan gunakan kembali aplikasi. Ini memungkinkan Amazon S3 klien layanan enkripsi sisi klien dalam aplikasi Anda untuk mendekripsi objek yang dibuat oleh klien layanan V2. Jika aplikasi Anda menggunakan beberapa AWS SDK, Anda harus memperbarui setiap SDK secara terpisah.
2. Migrasikan enkripsi dan dekripsi klien ke V2. Setelah semua klien enkripsi V1 Anda dapat membaca format enkripsi V2, perbarui enkripsi Amazon S3 sisi klien dan klien dekripsi dalam kode aplikasi Anda untuk menggunakan setara V2 mereka.

Perbarui Klien yang Ada untuk Membaca Format Baru

Klien enkripsi V2 menggunakan algoritma enkripsi yang AWS SDK for Java tidak didukung oleh versi lama.

Langkah pertama dalam migrasi adalah memperbarui klien enkripsi V1 Anda untuk menggunakan versi 1.11.837 atau yang lebih baru. AWS SDK for Java (Kami menyarankan Anda memperbarui ke versi rilis terbaru, yang dapat Anda temukan di [Referensi API Java versi 1.x.](#)) Untuk melakukannya, perbarui dependensi dalam konfigurasi proyek Anda. Setelah konfigurasi proyek Anda diperbarui, bangun kembali proyek Anda dan terapkan ulang.

Setelah Anda menyelesaikan langkah-langkah ini, klien enkripsi V1 aplikasi Anda akan dapat membaca objek yang ditulis oleh klien enkripsi V2.

Perbarui Ketergantungan dalam Konfigurasi Proyek Anda

Ubah file konfigurasi proyek Anda (misalnya, pom.xml atau build.gradle) untuk menggunakan versi 1.11.837 atau yang lebih baru. AWS SDK for Java Kemudian, bangun kembali proyek Anda dan gunakan kembali.

Menyelesaikan langkah ini sebelum menerapkan kode aplikasi baru membantu memastikan bahwa operasi enkripsi dan dekripsi tetap konsisten di seluruh armada Anda selama proses migrasi.

Contoh Menggunakan Maven

Cuplikan dari file pom.xml:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.11.837</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Contoh Menggunakan Gradle

Cuplikan dari file build.gradle:

```
dependencies {  
    implementation platform('com.amazonaws:aws-java-sdk-bom:1.11.837')  
    implementation 'com.amazonaws:aws-java-sdk-s3'  
}
```

Migrasi Klien Enkripsi dan Dekripsi ke V2

Setelah proyek Anda diperbarui dengan versi SDK terbaru, Anda dapat memodifikasi kode aplikasi Anda untuk menggunakan klien V2. Untuk melakukannya, perbarui kode Anda terlebih dahulu untuk menggunakan pembuat klien layanan baru. Kemudian berikan materi enkripsi menggunakan metode pada builder yang telah diganti namanya, dan konfigurasi klien layanan Anda lebih lanjut sesuai kebutuhan.

Cuplikan kode ini menunjukkan cara menggunakan enkripsi sisi klien dengan AWS SDK for Java, dan memberikan perbandingan antara klien enkripsi V1 dan V2.

V1

```
// minimal configuration in V1; default CryptoMode.EncryptionOnly.  
EncryptionMaterialsProvider encryptionMaterialsProvider = ...  
AmazonS3Encryption encryptionClient = AmazonS3EncryptionClient.encryptionBuilder()  
    .withEncryptionMaterials(encryptionMaterialsProvider)  
    .build();
```

V2

```
// minimal configuration in V2; default CryptoMode.StrictAuthenticatedEncryption.  
EncryptionMaterialsProvider encryptionMaterialsProvider = ...  
AmazonS3EncryptionV2 encryptionClient = AmazonS3EncryptionClientV2.encryptionBuilder()  
    .withEncryptionMaterialsProvider(encryptionMaterialsProvider)  
    .withCryptoConfiguration(new CryptoConfigurationV2()  
        // The following setting allows the client to read V1  
        encrypted objects  
        .withCryptoMode(CryptoMode.AuthenticatedEncryption)  
    )  
    .build();
```

Contoh di atas menetapkan `cryptoMode` ke `AuthenticatedEncryption`. Ini adalah pengaturan yang memungkinkan klien enkripsi V2 untuk membaca objek yang telah ditulis oleh klien enkripsi V1.

Jika klien Anda tidak memerlukan kemampuan untuk membaca objek yang ditulis oleh klien V1, maka sebaiknya gunakan pengaturan default `StrictAuthenticatedEncryption` sebagai gantinya.

Membangun Klien Enkripsi V2

Klien enkripsi V2 dapat dibangun dengan memanggil `AmazonS3 EncryptionClient v2.EncryptionBuilder ()`.

Anda dapat mengganti semua klien enkripsi V1 yang ada dengan klien enkripsi V2. Klien enkripsi V2 akan selalu dapat membaca objek apa pun yang telah ditulis oleh klien enkripsi V1 selama Anda mengizinkannya melakukannya dengan mengonfigurasi klien enkripsi V2 untuk menggunakan `AuthenticatedEncryption`cryptoMode`

Membuat klien enkripsi V2 baru sangat mirip dengan cara Anda membuat klien enkripsi V1. Namun, ada beberapa perbedaan:

- Anda akan menggunakan `CryptoConfigurationV2` objek untuk mengkonfigurasi klien alih-alih `CryptoConfiguration` objek. Parameter ini diperlukan.
- `cryptoMode` pengaturan default untuk klien enkripsi V2 adalah `StrictAuthenticatedEncryption`. Untuk klien enkripsi V1 itu `EncryptionOnly`.
- Metode `withEncryptionMaterials()` pada pembuat klien enkripsi telah diubah namanya menjadi `withEncryptionMaterialsProvider ()`. Ini hanyalah perubahan kosmetik yang lebih akurat mencerminkan jenis argumen. Anda harus menggunakan metode baru ketika Anda mengkonfigurasi klien layanan Anda.

Note

Saat mendekripsi dengan AES-GCM, baca seluruh objek sampai akhir sebelum Anda mulai menggunakan data yang didekripsi. Ini untuk memverifikasi bahwa objek belum dimodifikasi sejak dienkripsi.

Gunakan Penyedia Bahan Enkripsi

Anda dapat terus menggunakan penyedia bahan enkripsi yang sama dan objek materi enkripsi yang sudah Anda gunakan dengan klien enkripsi V1. Kelas-kelas ini bertanggung jawab untuk menyediakan kunci yang digunakan klien enkripsi untuk mengamankan data Anda. Mereka dapat digunakan secara bergantian dengan klien enkripsi V2 dan V1.

Konfigurasi Klien Enkripsi V2

Klien enkripsi V2 dikonfigurasi dengan `CryptoConfigurationV2` objek. Objek ini dapat dibangun dengan memanggil konstruktor defaultnya dan kemudian memodifikasi propertinya seperti yang diperlukan dari default.

Nilai default untuk `CryptoConfigurationV2` adalah:

- `cryptoMode = CryptoMode.StrictAuthenticatedEncryption`
- `storageMode = CryptoStorageMode.ObjectMetadata`
- `secureRandom =` contoh dari `SecureRandom`
- `rangeGetMode = CryptoRangeGetMode.DISABLED`
- `unsafeUndecryptableObjectPassthrough = false`

Perhatikan bahwa `EncryptionOnly` tidak didukung `cryptoMode` dalam klien enkripsi V2. Klien enkripsi V2 akan selalu mengenkripsi konten menggunakan enkripsi yang diautentikasi, dan melindungi kunci enkripsi konten (CEK) menggunakan objek V2. `KeyWrap`

Contoh berikut menunjukkan cara menentukan konfigurasi kriptografi di V1, dan cara membuat instance objek `CryptoConfigurationV2` untuk diteruskan ke pembuat klien enkripsi V2.

V1

```
CryptoConfiguration cryptoConfiguration = new CryptoConfiguration()
    .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

V2

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
    .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

Contoh Tambahan

Contoh berikut menunjukkan cara mengatasi kasus penggunaan tertentu yang terkait dengan migrasi dari V1 ke V2.

Konfigurasi Klien Layanan untuk Membaca Objek yang Dibuat oleh Klien Enkripsi V1

Untuk membaca objek yang sebelumnya ditulis menggunakan klien enkripsi V1, atur `cryptoMode` ke `AuthenticatedEncryption`. Cuplikan kode berikut menunjukkan bagaimana membangun objek konfigurasi dengan pengaturan ini.

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
    .withCryptoMode(CryptoMode.AuthenticatedEncryption);
```

Konfigurasi Klien Layanan untuk Mendapatkan Rentang Byte Objek

Untuk `get` dapat berbagai byte dari objek S3 terenkripsi, aktifkan pengaturan konfigurasi baru. `rangeGetMode` Pengaturan ini dinonaktifkan pada klien enkripsi V2 secara default. Perhatikan bahwa bahkan ketika diaktifkan, rentang `get` hanya berfungsi pada objek yang telah dienkripsi menggunakan algoritme yang didukung oleh `cryptoMode` pengaturan klien. Untuk informasi selengkapnya, lihat [CryptoRangeGetMode](#) di Referensi AWS SDK for Java API.

Jika Anda berencana untuk menggunakan Amazon S3 TransferManager untuk melakukan unduhan multipart dari Amazon S3 objek terenkripsi menggunakan klien enkripsi V2, maka Anda harus terlebih dahulu mengaktifkan `rangeGetMode` pengaturan pada klien enkripsi V2.

Cuplikan kode berikut menunjukkan cara mengkonfigurasi klien V2 untuk melakukan rentang. `get`

```
// Allows range gets using AES/CTR, for V2 encrypted objects only
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
    .withRangeGetMode(CryptoRangeGetMode.ALL);

// Allows range gets using AES/CTR and AES/CBC, for V1 and V2 objects
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
    .withCryptoMode(CryptoMode.AuthenticatedEncryption)
    .withRangeGetMode(CryptoRangeGetMode.ALL);
```

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/0H0VWHwQsaW
jMrGj000MFzXGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvvcMXnduLtkBEX7TISMPW+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.

Dokumentasi ini dibangun pada: 21 Mei 2024

21 Mei 2024, 2024

Hapus instruksi untuk mengatur properti `networkaddress.cache.ttl` keamanan dengan menggunakan properti sistem baris perintah java. Lihat, [Cara mengatur JVM TTL](#).

Januari 12, 2024

Tambahkan spanduk yang mengumumkan akhir dukungan untuk AWS SDK for Java v1.x.

6 Desember 2023

- Berikan kunci [OpenPGP saat ini](#).

14 Maret 2023

- Panduan yang diperbarui untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi selengkapnya, lihat [Praktik terbaik keamanan di IAM](#).

28 Juli 2022

- Menambahkan peringatan bahwa EC2-Classic akan pensiun pada 15 Agustus 2022.

Mar 22, 2018

- Menghapus mengelola sesi Tomcat sebagai DynamoDB contoh karena alat itu tidak lagi didukung.

Nov 2, 2017

- Menambahkan contoh kriptografi untuk klien Amazon S3 enkripsi, termasuk topik baru: [Gunakan enkripsi sisi Amazon S3 klien dan enkripsi sisi Amazon S3 klien dengan kunci terkelola AWS KMS dan enkripsi sisi klien dengan kunci master Amazon S3 klien](#).

14 Apr 2017

- Membuat sejumlah pembaruan pada AWS SDK for Java bagian [Amazon S3 Contoh Menggunakan](#), termasuk topik baru: [Mengelola Izin Amazon S3 Akses untuk Bucket dan Objek dan Mengonfigurasi Amazon S3 Bucket sebagai Situs Web](#).

04 Apr 2017

- Topik baru, [Mengaktifkan Metrik untuk AWS SDK for Java](#) menjelaskan cara menghasilkan metrik kinerja aplikasi dan SDK untuk AWS SDK for Java

03 Apr 2017

- Menambahkan CloudWatch contoh baru ke [CloudWatch Contoh Menggunakan AWS SDK for Java](#) bagian: [Mendapatkan Metrik dari CloudWatch](#), [Menerbitkan Data Metrik Kustom](#), [Bekerja dengan CloudWatch Alarm](#), [Menggunakan Tindakan Alarm di CloudWatch](#), dan [Mengirim Acara ke CloudWatch](#)

27 Mar 2017

- Menambahkan lebih banyak Amazon EC2 contoh ke AWS SDK for Java bagian [Amazon EC2 Contoh Menggunakan](#): [Mengelola Amazon EC2 Instans](#), [Menggunakan Alamat IP Elastis di Amazon EC2](#), [Menggunakan wilayah dan zona ketersediaan](#), [Bekerja dengan Pasangan Amazon EC2 Kunci](#), dan [Bekerja dengan Grup Keamanan di Amazon EC2](#).

21 Mar 2017

- Menambahkan serangkaian contoh IAM baru ke [Contoh IAM Menggunakan AWS SDK for Java](#) bagian: [Mengelola Kunci Akses IAM](#), [Mengelola Pengguna IAM](#), [Menggunakan Alias Akun IAM](#), [Bekerja dengan Kebijakan IAM](#), dan [Bekerja dengan Sertifikat Server IAM](#)

Mar 13, 2017

- Menambahkan tiga topik baru ke Amazon SQS bagian: [Mengaktifkan Polling Panjang untuk Antrian Amazon SQS Pesan](#), [Mengatur Batas Waktu Visibilitas Amazon SQS](#), dan [Menggunakan Antrian Surat Mati di Amazon SQS](#)

26 Jan 2017

- Menambahkan Amazon S3 topik baru, [Menggunakan TransferManager untuk Amazon S3 Operasi](#), dan [Praktik Terbaik untuk AWS Pengembangan baru dengan AWS SDK for Java](#) topik di [AWS SDK for Java bagian Menggunakan](#).

16 Jan 2017

- Menambahkan Amazon S3 topik baru, [Mengelola Akses ke Amazon S3 Bucket Menggunakan Kebijakan Bucket](#), dan dua Amazon SQS topik baru, [Bekerja dengan Antrian Amazon SQS Pesan](#) dan [Mengirim Menerima dan Menghapus Pesan Amazon SQS](#)

Desember 16, 2016

- Menambahkan contoh topik baru untuk DynamoDB: [Bekerja dengan Tabel di DynamoDB](#) dan [Bekerja dengan Item di DynamoDB](#).

September 26, 2016

- Topik di bagian Advanced telah dipindahkan ke [Menggunakan AWS SDK for Java](#), karena mereka benar-benar penting untuk menggunakan SDK.

25 Agustus 2016

- Topik baru, [Membuat Klien Layanan](#), telah ditambahkan ke [Menggunakan AWS SDK for Java](#), yang menunjukkan cara menggunakan pembangun klien untuk menyederhanakan pembuatan Layanan AWS klien.

Bagian [Contoh AWS SDK for Java Kode](#) telah diperbarui dengan [contoh baru untuk S3](#) yang didukung oleh [repositori GitHub](#) yang berisi kode contoh lengkap.

02 Mei 2016

- Topik baru, [Pemrograman Asinkron](#), telah ditambahkan ke AWS SDK for Java bagian [Menggunakan](#), menjelaskan cara bekerja dengan metode klien asinkron yang mengembalikan objek atau yang mengambil file. Future AsyncHandler

26 Apr 2016

- Topik Persyaratan Sertifikat SSL telah dihapus, karena tidak lagi relevan. Support untuk sertifikat yang ditandatangani SHA-1 tidak digunakan lagi pada tahun 2015 dan situs yang menyimpan skrip pengujian telah dihapus.

Mar 14, 2016

- Menambahkan topik baru ke Amazon SWF bagian: [Tugas Lambda](#), yang menjelaskan cara menerapkan Amazon SWF alur kerja yang memanggil Lambda fungsi sebagai tugas sebagai alternatif untuk menggunakan aktivitas tradisional. Amazon SWF

04 Mar 2016

- [Amazon SWF Contoh Menggunakan AWS SDK for Java bagian](#) telah diperbarui dengan konten baru:
 - [Amazon SWF Dasar-dasar](#) - Memberikan informasi dasar tentang cara memasukkan SWF dalam proyek Anda.
 - [Membangun Amazon SWF Aplikasi Sederhana](#) - Tutorial baru yang memberikan step-by-step panduan bagi pengembang Java yang baru Amazon SWF.
 - [Mematikan Aktivitas dan Alur Kerja Pekerja dengan Anggun](#) - Menjelaskan bagaimana Anda dapat menutup kelas Amazon SWF pekerja dengan anggun menggunakan kelas konkurensi Java.

Februari 23, 2016

- Sumber untuk Panduan AWS SDK for Java Pengembang telah dipindahkan ke [aws-java-developer-guide](#).


28 Des 2015

- [the section called “Mengatur JVM TTL untuk pencarian nama DNS”](#) telah dipindahkan dari Advanced ke [Using the AWS SDK for Java](#), dan telah ditulis ulang untuk kejelasan.

[Menggunakan SDK dengan Apache Maven](#) telah diperbarui dengan informasi tentang cara memasukkan bill of material (BOM) SDK dalam proyek Anda.

04 Agustus 2015

- Persyaratan Sertifikat SSL adalah topik baru di bagian [Memulai](#) yang menjelaskan AWS pindah ke sertifikat yang ditandatangani SHA256 untuk koneksi SSL, dan cara memperbaiki awal 1.6 dan lingkungan Java sebelumnya untuk menggunakan sertifikat ini, yang diperlukan untuk AWS akses setelah 30 September 2015.

 Note

Java 1.7+ sudah mampu bekerja dengan sertifikat yang ditandatangani SHA256.

Mei 14, 2014

- Materi [pengantar](#) dan [memulai](#) telah banyak direvisi untuk mendukung struktur panduan baru dan sekarang mencakup panduan tentang cara [Mengatur AWS Kredensial dan Wilayah](#) untuk Pembangunan.

Pembahasan [sampel kode](#) telah dipindahkan ke topiknya sendiri di bagian [Dokumentasi dan Sumber Daya Tambahan](#).

Informasi tentang cara [melihat riwayat revisi SDK](#) telah dipindahkan ke pendahuluan.

9 Mei 2014

- Struktur keseluruhan AWS SDK for Java dokumentasi telah disederhanakan, dan topik [Memulai](#) dan [Dokumentasi Tambahan dan Sumber Daya](#) telah diperbarui.

Topik baru telah ditambahkan:

- [Bekerja dengan AWS Credentials](#) - membahas berbagai cara yang dapat Anda tentukan kredensial untuk digunakan dengan AWS SDK for Java
- [Menggunakan Peran IAM untuk Memberikan Akses ke AWS Sumber Daya pada Amazon EC2](#) - memberikan informasi tentang cara menentukan kredensial dengan aman untuk aplikasi yang berjalan pada instans EC2.

September 9, 2013

- Topik ini, Riwayat Dokumen, melacak perubahan pada Panduan AWS SDK for Java Pengembang. Hal ini dimaksudkan sebagai pendamping sejarah catatan rilis.