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

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 Javamenyediakan API Java untukAWS layanan. Menggunakan SDK, Anda dapat dengan mudah membangun aplikasi Java yang bekerja denganAmazon S3,,Amazon EC2DynamoDB, dan banyak lagi. Kami secara teratur menambahkan dukungan untuk layanan baru keAWS 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

LihatlahAWS SDK for Java 2.x baru di https://github.com/aws/aws-sdk-java-v 2/. Ini mencakup banyak fitur yang ditunggu, seperti cara untuk mencolokkan implementasi HTTP. Untuk memulai, lihat AWS SDK for Java2.x Developer Guide.

Dokumentasi dan Sumber Daya Tambahan

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

- AWS SDK for JavaReferensi 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 (Kericau)
- catatan rilis

Eclipse Support

Jika Anda mengembangkan kode menggunakan Eclipse IDE, Anda dapat menggunakan AWS Toolkit for Eclipseuntuk menambahkanAWS SDK for Java ke proyek Eclipse yang ada atau untuk membuatAWS SDK for Java proyek baru. Toolkit ini juga mendukung pembuatan dan pengunggahanLambda fungsi, peluncuran dan pemantauanAmazon EC2 instans, mengelolaIAM pengguna dan grup keamanan, editorAWS CloudFormation template, dan banyak lagi.

Lihat PanduanAWS 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 (AWSMobile SDK for Android).

Melihat Riwayat Revisi SDK

Untuk melihat riwayat rilisAWS SDK for Java, termasuk perubahan dan layanan yang didukung per versi SDK, lihat catatan rilis SDK.

Membangun Dokumentasi Referensi Java untuk versi SDK Sebelumnya

ReferensiAWS 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 SDKGitHub.
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 menggunakanAWS SDK for Java.

Topik

Pengaturan dasar untuk bekerja dengan Layanan AWS

Gambaran Umum

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

- Anda harus dapat masuk ke portal AWS akses yang tersedia diAWS 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 PowerUserAccessAWSdikelola cukup untuk sebagian besar kebutuhan pengembangan.
- Lingkungan pengembangan dengan elemen berikut:
  - File konfigurasi bersama yang disiapkan dengan cara berikut:
    - `config`File berisi profil default yang menentukan file. Wilayah AWS
    - `credentials`File berisi kredensil 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 denganLayanan AWS. AWS Toolkit for IntelliJDan AWS Toolkit for Eclipsedua 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 kredensil 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. `json`Pengaturan 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, konfigurasikan 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 kredenial 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 kredenial Anda opsi dan salin isinya.

6. Tempel kredensi yang disalin ke `credentials` file lokal Anda dan hapus nama profil apa pun yang disisipkan. File Anda akan menyerupai berikut ini:

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

7. Simpan `credentials` file

SDK for Java akan mengakses kredensil sementara ini ketika membuat klien layanan dan menggunakannya untuk setiap permintaan. Pengaturan untuk peran IAM yang dipilih pada langkah 5a menentukan berapa lama kredensil sementara valid. Durasi maksimum adalah dua belas jam.

Setelah kredensi sementara berakhir, ulangi langkah 4 hingga 7.

Instal Lingkungan Pengembangan Java

AWS SDK for Javamemerlukan J2SE Development Kit 6.0 atau yang lebih baru. Anda dapat mengunduh perangkat lunak Java terbaru dari http://www.oracle.com/technetwork/java/javase/downloads/.

> **Important**
>
> Versi Java 1.6 (JS2E 6.0) tidak memiliki dukungan bawaan untuk sertifikat SSL yang ditandatangani SHA256, yang diperlukan untuk semua koneksi HTTPS setelah 30 September 2015. AWS
> Versi Java 1.7 atau yang lebih baru dikemas dengan sertifikat yang diperbarui dan tidak terpengaruh oleh masalah ini.

Memilih JVM

Untuk kinerja terbaik dari aplikasi berbasis server Anda denganAWS SDK for Java, kami sarankan Anda menggunakan versi 64-bit dari Java Virtual Machine (JVM). JVM ini hanya berjalan dalam mode server, bahkan jika Anda menentukan `-Client` opsi pada waktu berjalan.

Menggunakan versi 32-bit dari JVM dengan `-Server` opsi pada waktu berjalan harus memberikan kinerja yang sebanding dengan JVM 64-bit.

Cara untuk mendapatkanAWS SDK for Java

Prasyarat

Untuk menggunakan layananAWS SDK for Java, Anda harus memiliki:

- Anda harus dapat masuk ke portalAWS akses yang tersedia diAWS IAM Identity Center.
- Instalasi Java yang cocok.
- Kredensyal sementara diatur dalam`credentials` file bersama lokal Anda.

Lihatthe section called “Pengaturan dasar” topik untuk petunjuk tentang cara menyiapkan untuk menggunakan SDK for Java.

Gunakan alat build untuk mengelola dependensi SDK for Java

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 SDK versi sebelumnya, lihat Menginstal SDK versi sebelumnya.

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

SDK berisi direktori berikut:

- `documentation`- berisi dokumentasi API (juga tersedia di web: ReferensiAWS SDK for Java API).
- `lib`- berisi`.jar` file SDK.

- `samples`- berisi kode contoh kerja yang menunjukkan bagaimana 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 `kelib` dan`third-party` direktori ke dependensi dalam file build Anda, dan tambahkan ke java Anda`CLASSPATH` 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 men-download 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. Sebagai contoh, `1.6.10`.
3. Klik tombol Unduh ZIP untuk mengunduh versi SDK yang Anda pilih.
4. Unzip file ke direktori pada sistem pengembangan Anda. Pada banyak sistem, Anda dapat menggunakan pengelola file grafis Anda untuk melakukan ini, atau menggunakan`unzip` utilitas di jendela terminal.
5. Di jendela terminal, navigasikan ke direktori tempat Anda membuka ritsleting sumber SDK.
6. Membangun dan menginstal SDK dengan perintah berikut (Maven required):

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

   `.jar`File yang dihasilkan dibangun ke dalam`target` direktori.
7. (Opsional) Bangun 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

Anda dapat menggunakan Apache Maven untuk mengkonfigurasi dan membangunAWS 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 menggunakanAWS 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, gunakanAWS 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 kepom.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>
```

Untuk melihat versi terbaruAWS 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 AWSuntuk mempelajari apa dependensi untuk digunakan untuk diberikanLayanan AWS. Lihat file POM di bawah contoh layanan tertentu. Misalnya, jika Anda tertarik dengan dependensi untuk layananAWS 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 mendeklarasikannyapom.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`0jar` file Anda di`target` 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 untukGradleproyek, impor Maven BOM untukAWS SDK for Javake dalam aplikasi`build.gradle`berkas.

> **Note**
>
> Dalam contoh berikut, ganti*`1.12.529`*dalam file build dengan versi yang valid dariAWS SDK for Java. Temukan versi terbaru diRepositori 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, aktifkanDITINGKATKAN_POM_SUPPORTfitur di`settings.gradle`berkas.

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

2. Tambahkan BOM kedependensibagian 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 didependensibagian. Misalnya, berikut ini mencakup ketergantungan untukAmazon 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 untukAmazon 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 untukAmazon S3dengan dependensi dariAWSlayanan yang akan Anda gunakan dalam proyek Anda. Modul (dependensi) yang dikelola olehAWS SDK for JavaBOM terdaftar diRepositori 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 mengelolaAWS SDK for Javadependensi untuk proyek Anda, gunakan Springplugin manajemen ketergantunganagar Gradle mengimpor Maven BOM untuk SDK.

1. Tambahkan plugin manajemen ketergantungan ke aplikasi Andabuild.gradleberkas.

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

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

2. Tambahkan BOM keManajemen Ketergantunganbagian dari file.

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

3. Tentukan modul SDK yang akan Anda gunakan didependensibagian. Misalnya, berikut ini mencakup ketergantungan untukAmazon 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`file yang menyertakan ketergantungan untukAmazon 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 untukAmazon S3dengan dependensi dariAWSlayanan yang akan Anda gunakan dalam proyek Anda. Modul (dependensi) yang dikelola olehAWS SDK for JavaBOM terdaftar diRepositori pusat Maven.

Untuk informasi selengkapnya tentang menentukan dependensi SDK dengan menggunakan BOM, lihatMenggunakan SDK dengan Apache Maven.

Menyiapkan kredensyalAWS sementara danWilayah AWS untuk pengembangan

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

Topik ini memberikan informasi dasar tentang menyiapkan kredensyalAWS sementara Anda untuk pengembangan aplikasi lokal menggunakanAWS SDK for Java. Jika Anda perlu menyiapkan kredensyal 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 keAWS Sumber Daya aktifAmazon EC2.
- MengaturAWS kredensyal dalam Eclipse menggunakan AWS Toolkit for Eclipse. Lihat MengaturAWS Kredensyal di PanduanAWS Toolkit for Eclipse Pengguna untuk informasi selengkapnya.

Konfigurasikan kredenah sementara

Anda dapat mengkonfigurasi kredensyal sementara untukAWS SDK for Java dalam beberapa cara, tetapi di sini adalah pendekatan yang disarankan:

- Tetapkan kredensyal sementara di file profilAWS kredensyal 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

  Lihatthe section called “Menyiapkan kredensi sementara untuk SDK” di panduan ini untuk petunjuk tentang cara mendapatkan kredensyal sementara Anda.

- MengaturAWS_ACCESS_KEY_ID,AWS_SECRET_ACCESS_KEY, dan variabelAWS_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 IAMAmazon EC2 di PanduanAmazon EC2 Pengguna untuk Instans Linux untuk diskusi mendetail tentang cara kerjanya.

Setelah Anda menetapkan kredensyalAWS sementara Anda menggunakan salah satu metode ini, mereka akan dimuat secara otomatis olehAWS SDK for Java dengan menggunakan rantai penyedia kredensi default. Untuk informasi lebih lanjut tentang bekerja denganAWS kredensyal dalam aplikasi Java Anda, lihat Bekerja denganAWS Kredensyal.

Kredenah IMDS yang menyegarkan

AWS SDK for JavaMendukung keikutsertaan kredensyal IMDS yang menyegarkan di latar belakang setiap 1 menit, terlepas dari waktu kedaluwarsa kredensialnya. Ini memungkinkan Anda untuk menyegarkan kredensyal lebih sering dan mengurangi kemungkinan bahwa tidak mencapai IMDS memengaruhiAWS 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();
```

MengaturWilayah AWS

Anda harus menetapkan defaultWilayah AWS yang akan digunakan untuk mengaksesAWS layanan denganAWS 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 ReferensiAmazon 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 kredensyal untuk mengaturAWS wilayah default Anda:

- Atur fileAWS konfigurasiWilayah 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 inginkanWilayah 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 adalahWilayah 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

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 ResetExceptionketika 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 ResetExceptionadalah 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 FileInputStreamtetapi 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 untuk DynamoDB bernama AmazonDynamoDB ClientBuilder.

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 builder, Anda dapat menyesuaikan properti klien dengan menggunakan banyak setter fasih di API builder. Misalnya, Anda dapat mengatur wilayah kustom dan penyedia kredensyal 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 ClientConfigurationdan 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 ExecutorFactoryuntuk 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 meneleponshutdown, 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 EnvironmentVariableCredentialsProviderkelas untuk memuat kredensyal ini.
2. Properti sistem Java -`aws.accessKeyId`,`aws.secretKey`, dan`aws.sessionToken`. AWS SDK for Java Penggunaan SystemPropertiesCredentialsProvideruntuk 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 Penggunaan ProfileCredentialsProvideruntuk 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 Penggunaan ContainerCredentialsProvideruntuk 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 Penggunaan InstanceProfileCredentialsProvideruntuk 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 kredensyal sementara

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

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

Menetapkan profil kredensyal alternatif

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

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 kredensyal 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 kredensyal alternatif

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

Anda dapat menggunakan fitur ini untuk sementara mengubah lokasi di mana AWS SDK for Java mencari file kredensyal 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 kredensyal default

- Atur variabel `AWS_CREDENTIAL_PROFILES_FILE` lingkungan ke lokasi file AWS kredensyal 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 kredensyal 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, sepertimetadata_service_timeout, danmetadata_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 kredensyal

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

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

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

Tentukan penyedia kredensyal atau rantai penyedia

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

Anda memberikan instance penyedia kredensyal atau rantai penyedia ke pembuat klien yang mengambil AWSCredentialsProviderantarmuka sebagai input. Contoh berikut menunjukkan cara menggunakan kredensyal lingkungan secara khusus.

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

Untuk daftar lengkap penyedia kredensyal 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 kredensyal Anda sendiri yang mengimplementasikan `AWSCredentialsProvider` antarmuka, atau dengan mensubklasifikasikan kelas. AWSCredentialsProviderChain

Secara eksplisit menentukan kredensyal sementara

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

1. Buat instance BasicSessionCredentialskelas, dan berikan kunci AWS akses, kunci AWS rahasia, dan token AWS sesi yang akan digunakan SDK untuk koneksi.
2. Buat AWSStaticCredentialsProviderdengan `AWSCredentials` objek.
3. Konfigurasikan pembuat klien dengan `AWSStaticCredentialsProvider` dan bangun klien.

Berikut sebuah contoh.

```
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 Kredensyal dan Wilayah untuk Pembangunan
- Menggunakan Peran IAM untuk Memberikan Akses ke AWS Sumber Daya 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. Setiap awalan titik akhir layanan didefinisikan dalam antarmuka layanan. Misalnya, awalan DynamoDB endpoint didefinisikan dalam AmazonDynamo DB.

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, kami sarankan Anda menggunakan 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 pembuat 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 Kredenal 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 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)

AmazonServiceExceptionadalah 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, subkelas 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

AmazonClientExceptionmenunjukkan bahwa masalah terjadi di dalam kode klien Java, baik saat mencoba mengirim permintaan ke AWS atau saat mencoba mengurai respons dari AWS. An `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: on`Success` dan on`Error`.

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` dieksekusi 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 ExecutorServicedan 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 ThreadFactoryuntuk mengontrol cara nama thread di pool, atau untuk mencatat informasi tambahan tentang penggunaan thread.

Akses Asinkron

TransferManagerKelas 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 dari 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:
q963XH63Lcovl5Rr71APlzlye99rmWwT9DfuQaNznkD, ) 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-0000000000000, AWSAccessKeyId: ACCESSKEYID,
Version: 2009-10-16, SignatureVersion: 2, Timestamp:
2009-12-17T17:53:04.467Z, Signature:
9WcgfPwTobvLVcpyhbrdN7P7l3uH0oviYQ4yZ+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 dari 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 ClientConfigurationobjek 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 ClientConfigurationobjek. Opsi baru terkadang 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. setMaxConnectionsmetode.

> **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. setConnectionTimeoutmetode.
- 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, gunakanClientConfiguration.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 tidak lebih dari 60 detik. Ini memastikan bahwa ketika alamat IP sumber daya berubah, aplikasi Anda akan dapat menerima dan menggunakan alamat IP baru sumber daya dengan meminta DNS.

Pada beberapa konfigurasi Java, TTL default JVM diatur sehingga tidak akan pernah menyegarkan entri DNS sampai JVM dimulai ulang. Jadi, jika alamat IP untuk AWS sumber daya berubah saat aplikasi Anda masih berjalan, itu tidak akan dapat menggunakan sumber daya itu sampai Anda secara manual me-restart JVM dan informasi IP cache di-refresh. Dalam hal ini, sangat penting untuk mengatur TTL JVM sehingga secara berkala akan menyegarkan informasi IP cache.

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

Cara Mengatur JVM TTL

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

- Secara global, untuk semua aplikasi yang menggunakan JVM. Setel `networkaddress.cache.ttl` dalam `$JAVA_HOME/jre/lib/security/java.security` file untuk Java 8 atau `$JAVA_HOME/conf/security/java.security` file untuk Java 11 atau lebih tinggi:

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

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

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

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 kredensyal 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 kredensyal. 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 wilayah CloudWatch 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}
```

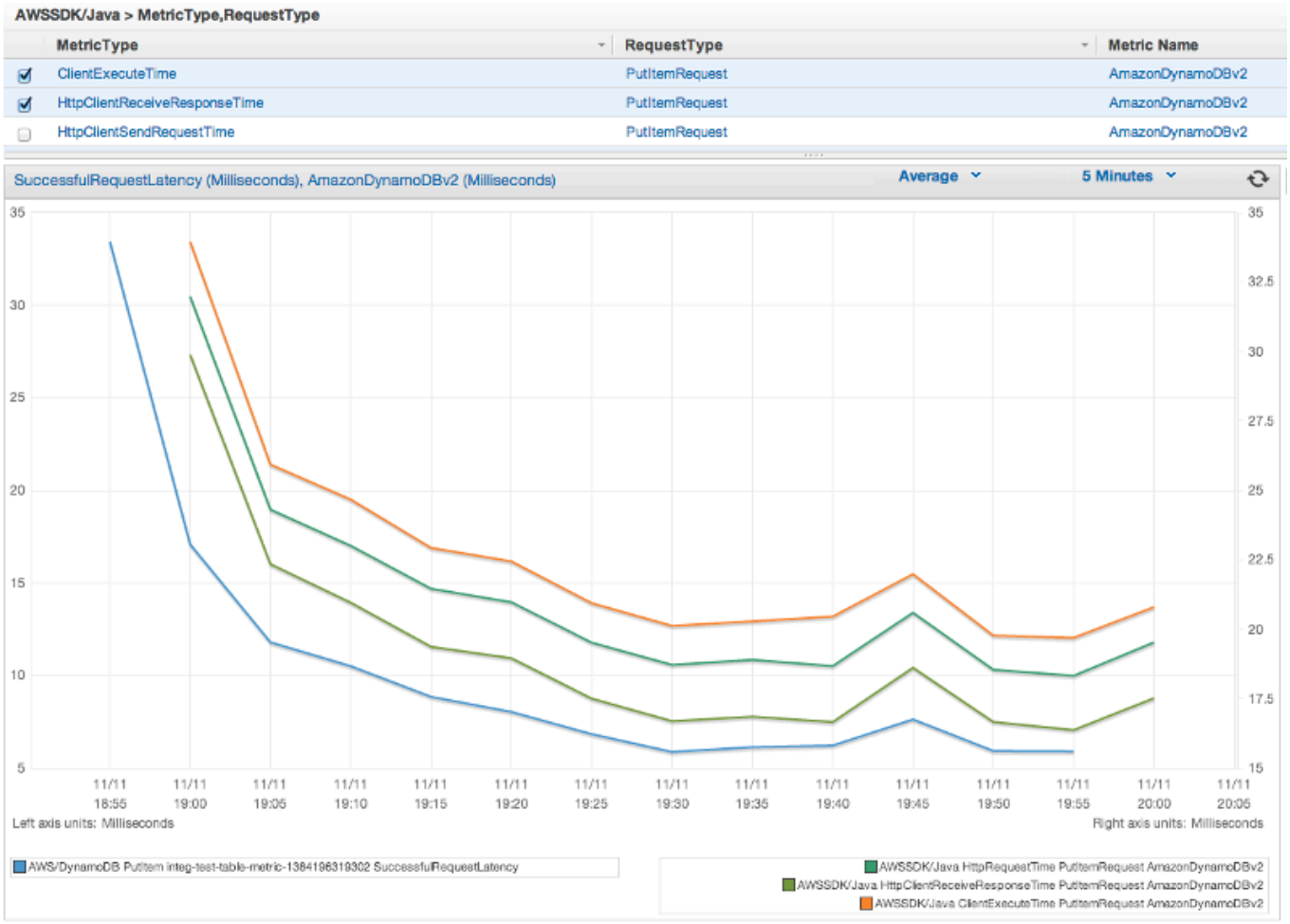
Setelah Anda mengaktifkan fitur, setiap kali ada permintaan layanan AWS dari, titik data metrik akan dibuat AWS SDK for Java, antri untuk ringkasan statistik, dan diunggah secara asinkron menjadi sekitar sekali setiap menit. CloudWatch Setelah metrik diunggah, Anda dapat memvisualisasikannya menggunakan AWS Management Consoledan 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:

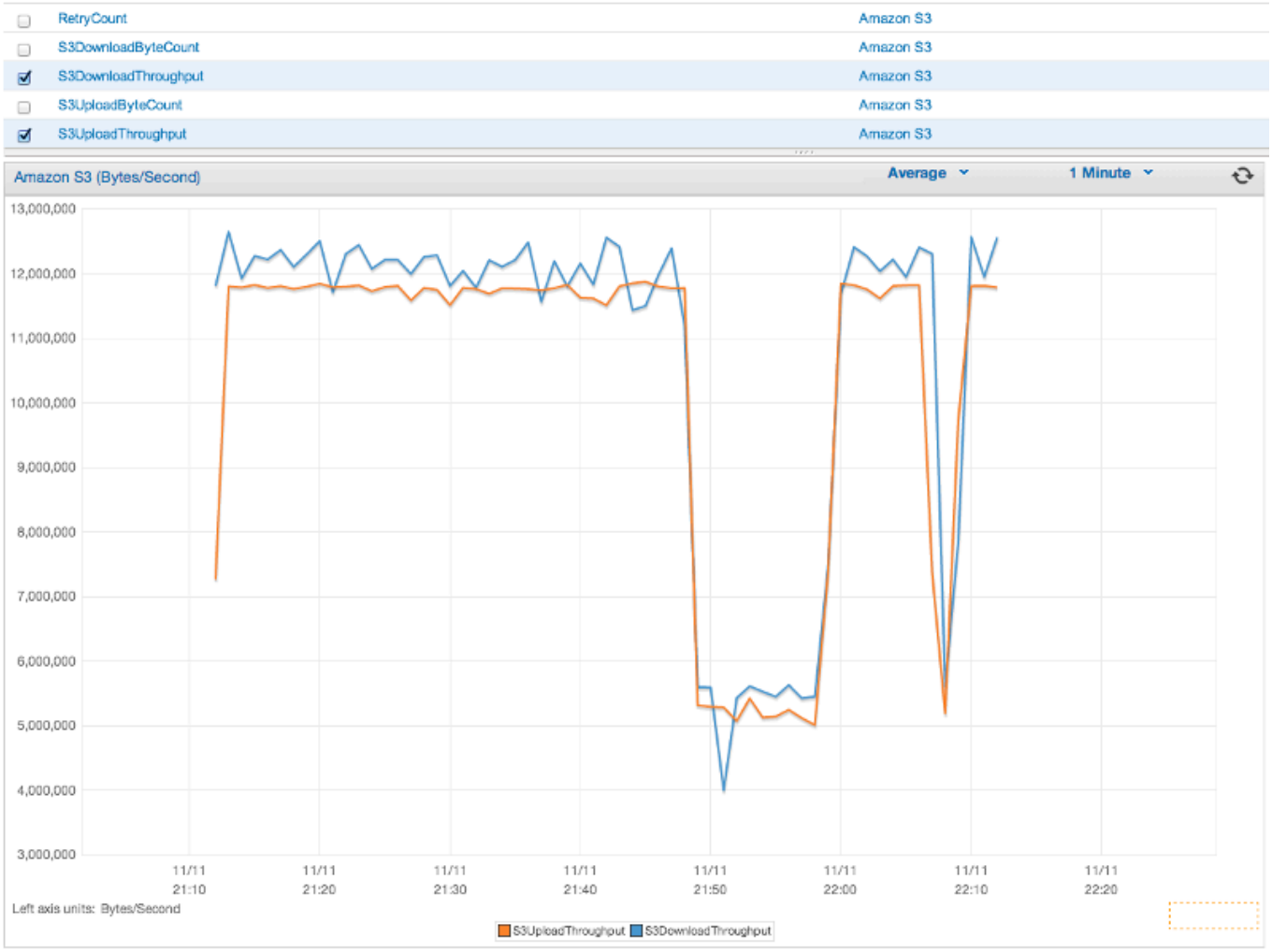
AWS Minta Metrik

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



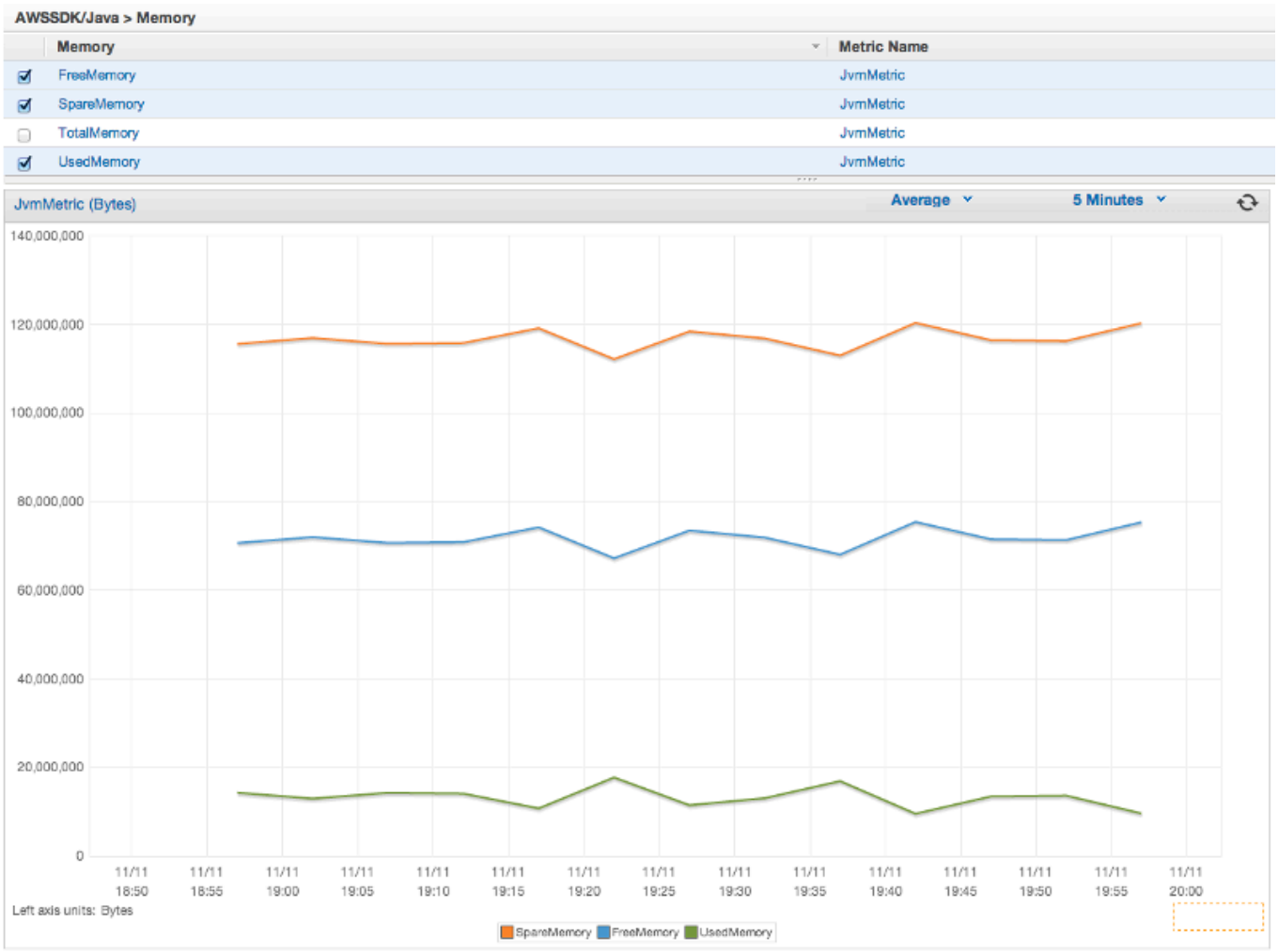
Layanan AWS Metrik

- 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 respository 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 pemrogramanCloudWatchmenggunakanAWS 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. Parameterkode 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

Metrik Listing

Daftar CloudWatch metrik, membuatListMetricsRequestdan 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 olehAWSlayanan dapat ditemukan dalam {https —docs-aws-amazon-com-amazoncloudwatch-terbaru-monitoring-cw-support-for-AWS-html} [Amazon CloudWatch Referensi Metrik dan Dimensi] diAmazon CloudWatchPanduan 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 dalamListMetricsResultdengan memanggilgetMetricsmetode. Hasilnya mungkinpaged. Untuk mengambil batch berikutnya hasil, hubungisetNextTokenpada objek permintaan asli dengan nilai kembali dariListMetricsResultobjekgetNextTokenmetode, dan lulus objek permintaan dimodifikasi kembali ke panggilan lain untuklistMetrics.

Informasi Selengkapnya

- ListMetricsdiAmazon CloudWatchReferensi API.

Memublikasikan Data Metrik Khusus

SejumlahAWSmemublikasikan layananMetrik mereka sendiridi ruang nama yang dimulai dengan”AWS “Anda juga dapat mempublikasikan data metrik kustom menggunakan namespace Anda sendiri (asalkan tidak dimulai dengan”AWS “).

Publikasikan Data Metrik Kustom

Untuk mempublikasikan data metrik Anda sendiri, hubungi AmazonCloudWatchClient'sputMetricDataMetode denganPutmetricDataRequest. ParameterPutMetricDataRequestharus menyertakan namespace khusus untuk digunakan untuk data, dan informasi tentang titik data itu sendiri dalamMetricDatumobjek.

> **Note**
>
> Anda tidak dapat menentukan namespace yang dimulai dengan”AWS “. Ruang nama yang dimulai dengan”AWS “disimpan untuk digunakan olehAmazon Web Servicesproduk.

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

- MenggunakanAmazon CloudWatchMetrikdi dalamAmazon CloudWatchPanduan Pengguna.
- AWSNamespacedi dalamAmazon CloudWatchPanduan Pengguna.
- PutMetricDatadi dalamAmazon CloudWatchReferensi API.

Bekerja dengan CloudWatch Alarm

Buat alarm

Untuk membuat alarm didasarkan pada CloudWatch metrik, panggil AmazonCloudWatchClient`putMetricAlarm`Metode denganPutMetricAlarmRequestdiisi 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`padaDescribeAlarmsResultyang dikembalikan oleh`describeAlarms`.

Hasilnya mungkinpaged. 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 denganDeleteAlarmsRequestberisi 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

- MembuatAmazon CloudWatchAlarmdiAmazon CloudWatchPanduan Pengguna
- PutMetricAlarmdiAmazon CloudWatchReferensi API
- DescribeAlarmsdiAmazon CloudWatchReferensi API
- DeleteAlarmsdiAmazon CloudWatchReferensi API

Menggunakan Tindakan Alarm di CloudWatch

Menggunakan CloudWatch Tindakan alarm, Anda dapat membuat alarm yang melakukan tindakan seperti secara otomatis menghentikan, mengakhiri, menyalakan ulangAmazon EC2contoh.

> **Note**
>
> Tindakan alarm dapat ditambahkan ke alarm dengan menggunakanPutMetricAlarmRequest's `setAlarmActions`Metode ketikamembuat alarm.

Mengaktifkan Tindakan Alarm

Untuk mengaktifkan tindakan alarm untuk CloudWatch alarm, panggil AmazonCloudWatchClient`enableAlarmActions`denganEnableAlarmActionsRequestberisi 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 Contohdi dalamAmazon CloudWatchPanduan Pengguna

- PutMetricAlarmdi dalamAmazon CloudWatchReferensi API
- EnableAlarmActionsdi dalamAmazon CloudWatchReferensi API
- DisableAlarmActionsdi dalamAmazon CloudWatchReferensi API

Mengirim Peristiwa keCloudWatch

CloudWatchPeristiwa memberikan aliran kejadian sistem secara hampir waktu-nyata yang menjelaskan perubahan dalamAWSsumber daya untukAmazon EC2contoh,Lambdafungsi,Kinesisaliran,Amazon ECStugas,Step Functionsmesin status,Amazon SNStopik,Amazon SQSantrian, atau target bawaan. Anda dapat mencocokkan peristiwa dan merutekannya ke satu atau beberapa fungsi atau pengaliran target dengan menggunakan aturan sederhana.

Tambahkan Acara

Untuk menambahkan kustomCloudWatchperistiwa, memanggil AmazonCloudWatcheventsClient`putEvents`dengan metodePutEventsRequestobjek yang berisi satu atau beberapaPutEventsRequestEntryobjek 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 AmazonCloudWatchVentsClient'sputRuledengan metodePutRuleRequestdengan nama aturan dan parameter opsional sepertiPola peristiwa,IAMperan untuk mengasosiasikan dengan aturan, danEkspresi penjadwalkanyang 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 termasukAmazon EC2contoh,Lambdafungsi,Kinesisaliran,Amazon ECStugas,Step Functionsmesin negara, dan target built-in.

Untuk menambahkan target ke aturan, panggil AmazonCloudWatchVentsClient'`sputTargets`dengan metodePutTargetsRequestberisi 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 PutEventsdiAmazon CloudWatch EventsPanduan Pengguna
- Ekspresi Jadwal untuk AturandiAmazon CloudWatch EventsPanduan Pengguna
- Jenis Peristiwa untukCloudWatchPeristiwadiAmazon CloudWatch EventsPanduan Pengguna
- Pola Peristiwa dan PeristiwadiAmazon CloudWatch EventsPanduan Pengguna

- PutEventsdiAmazon CloudWatch EventsReferensi API
- PutTargetsdiAmazon CloudWatch EventsReferensi API
- PutRulediAmazon CloudWatch EventsReferensi API

DynamoDBContoh MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrogramanDynamoDBmenggunakanAWS SDK for Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parametercontoh 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 denganDynamoDB
- Cara Menggunakan ItemDynamoDB

Cara dengan denganDynamoDB

Tabel adalah wadah untuk semua item dalamDynamoDB database. Sebelum Anda dapat melakukan tambahkan atau hapus data dariDynamoDB, 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 ScalarAttributeTypekelas. Nilai kunci dapat biner (B), numerik (N), atau string (S). Untuk informasi selengkapnya, lihat Aturan Penamaan dan Jenis Data di PanduanAmazon DynamoDB Pengembang.

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

> **Note**
>
> Amazon DynamoDBharga 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 membuatDynamoDB 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 membuatDynamoDB tabel Anda.

> **Note**
>
> Jika tabel dengan nama yang Anda pilih sudah ada, sebuah AmazonServiceExceptiondilemparkan.

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 lengkapnyaGitHub.

Buat Tabel dengan Kunci Utama Komposit

Tambahkan yang lain AttributeDefinitiondan KeySchemaElementke 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 lengkapnyaGitHub.

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 ResourceNotFoundExceptiondilemparkan.

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 ListTablesResultobjek 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 lengkapnyaGitHub.

Jelaskan (Dapatkan Informasi tentang) Tabel

Panggil`describeTable` metode DynamoDBklien.

> **Note**
>
> Jika tabel bernama tidak ada untuk akun dan wilayah Anda, sebuah ResourceNotFoundExceptiondilemparkan.

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.getTableName());
        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 lengkapnyaGitHub.

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 ResourceNotFoundExceptiondilemparkan.

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.getErrorMessage());
    System.exit(1);
}
```

Lihat contoh lengkapnyaGitHub.

Menghapus Tabel

Panggil`deleteTable` metode DynamoDBklien dan berikan nama tabel.

> **Note**
>
> Jika tabel bernama tidak ada untuk akun dan wilayah Anda, sebuah ResourceNotFoundExceptiondilemparkan.

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.getErrorMessage());
    System.exit(1);
}
```

Lihat contoh lengkapnyaGitHub.

Info Selengkapnya

- Pedoman untuk Bekerja dengan Tabel di PanduanAmazon DynamoDB Pengembang
- Bekerja dengan TabelDynamoDB di dalam PanduanAmazon DynamoDB Pengembang

Cara Menggunakan ItemDynamoDB

MasukDynamoDB, item adalah koleksiatribut, yang masing-masing memilikinamadan anilai. Nilai atribut dapat berupa skalar, set, atau jenis dokumen. Untuk informasi selengkapnya, lihatPeraturan Penamaan dan Jenis DatadiAmazon DynamoDBPanduan Developer.

Ambil (Dapatkan) Item dari Tabel

Panggil AmazonDynamoDB`getItem`dan menyebarkannyaGetItemRequestobjek dengan nama tabel dan nilai kunci primer dari item yang Anda inginkan. Ia mengembalikanGetItemResultobjek.

Anda dapat menggunakan kembali`GetItemResult`objek`getItem()`metode untuk mengambilPetakunci (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.getErrorMessage());
    System.exit(1);
```

LihatLengkapi Contohdi GitHub.

Tambahkan Item Baru ke Tabel

BuatPetapasangan 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, bidangnyadiperbaruioleh permintaan.

> **Note**
>
> Jika tabel bernama tidak ada untuk akun dan wilayah Anda, aPengecualianSumberDayaTidakDitemukandilemparkan.

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);
```

LihatLengkapi Contohdi 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, aPengecualianSumberDayaTidakDitemukandilemparkan.

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);
```

LihatLengkapi Contohdi GitHub.

Menggunakan class DynamoDBMapper

ParameterAWS SDK for Javamenyediakan`DynamoDBMapper`class, yang memungkinkan Anda memetakan kelas sisi klienAmazon DynamoDBtabel. Untuk menggunakan`DynamoDBMapper`kelas, Anda mendefinisikan hubungan antara item dalamDynamoDBtabel 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 menggunakanDynamoDBMapperkelas. Setelah konten ditambahkan ke tabel, perhatikan bahwa item dimuat dengan menggunakanPartisidanUrutkankunci. makaPenghargaanitem diperbarui. Untuk informasi tentang membuatMusikmeja, lihatBuat TabeldiAmazon 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.getStackTrace();
        }
    }

    @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;
        }
    }
```

LihatLengkapi Contohdi GitHub.

Info Selengkapnya

- Pedoman Cara Menggunakan ItemdiAmazon DynamoDBPanduan Pengembang
- Cara Menggunakan ItemDynamoDBdiAmazon DynamoDBPanduan Pengembang

Amazon EC2Contoh MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrogramanAmazon EC2denganAWS SDK for Java.

Topik

- Tutorial: Memulai instans EC2
- Menggunakan Peran IAM untuk Memberikan Akses keAWSSumber Daya diAmazon EC2
- Tutorial:Amazon EC2 Instans Spot
- Tutorial: LanjutanAmazon EC2Manajemen permintaan spot
- MengelolaAmazon EC2Instans
- Menggunakan Alamat Elastic IPAmazon EC2
- Gunakan wilayah dan availability zone
- Bekerja denganAmazon EC2Pasangan Kunci
- Bekerja dengan GrupAmazon EC2

Tutorial: Memulai instans EC2

Tutorial ini menunjukkan bagaimana menggunakanAWS SDK for Javauntuk memulai instans EC2.

Topik

Prasyarat

Sebelum memulai, pastikan bahwa Anda telah membuatAkun AWSdan bahwa Anda telah mengaturAWSkredensi. Untuk informasi selengkapnya, lihatMemulai.

BuatAmazon EC2Grup keamanan

EC2-Classic pensiun

> **Warning**
>
> Kami pensiun EC2-Classic pada 15 Agustus 2022. Kami menyarankan Anda untuk bermigrasi dari EC2-Classic ke VPC. Untuk informasi selengkapnya, lihatMemindahkan dari EC2-Classic ke VPCdi dalamPanduan Pengguna Amazon EC2 untuk Instans LinuxatauPanduan Pengguna Amazon EC2 untuk Instans Windows. Lihat juga posting blogEC2-Classic-Classic Networking Pensiun - Inilah Cara Mempersiapkan.

Buatgrup keamanan, yang bertindak sebagai firewall yang mengontrol lalu lintas jaringan untuk satu atau lebih instans EC2. Secara defaultAmazon EC2mengaitkan instans Anda dengan grup keamanan yang tidak memungkinkan lalu lintas masuk. Anda dapat membuat grup keamanan yang memungkinkan instans EC2 Anda menerima lalu lintas tertentu. Misalnya, jika Anda perlu terhubung ke instans Linux, Anda harus mengkonfigurasi grup keamanan untuk mengizinkan lalu lintas SSH. Anda dapat membuat grup keamanan menggunakanAmazon EC2konsolAWS SDK for Java.

Anda membuat grup keamanan untuk digunakan di EC2-Classic atau EC2-VPC. Untuk informasi selengkapnya tentang EC2-Classic dan EC2-VPC, lihatPlatform yang Didukungdi dalamAmazon EC2Panduan Pengguna untuk Instans Linux.

Untuk informasi selengkapnya tentang cara membuat grup keamanan menggunakanAmazon EC2konsol, lihatAmazon EC2Kelompok Keamanandi dalamAmazon EC2Panduan Pengguna untuk Instans Linux.

1. Membuat dan menginisialisasiCreateSecurityGroupRequestmisalnya. GunakanwithGroupNamemetode untuk mengatur nama grup keamanan, danwithDescriptionmetode untuk mengatur deskripsi grup keamanan, sebagai berikut:

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

   Nama grup keamanan harus unik diAWSwilayah di mana Anda menginisialisasiAmazon EC2klien Anda harus menggunakan karakter US-ASCII untuk nama dan deskripsi grup keamanan.

2. Lulus objek permintaan sebagai parameter kecreateSecurityGroupmetode Metode mengembalikanCreateSecurityGroupResultobjek, sebagai berikut:

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

   Jika Anda mencoba membuat grup keamanan dengan nama yang sama dengan grup keamanan yang ada,`createSecurityGroup`melempar pengecualian.

Secara default, grup keamanan baru tidak mengizinkan lalu lintas masuk ke AndaAmazon EC2misalnya. Untuk mengizinkan lalu lintas masuk, Anda harus secara eksplisit mengotorisasi masuknya grup keamanan. Anda dapat mengotorisasi ingress untuk masing-masing alamat IP, untuk berbagai alamat IP, untuk protokol tertentu, dan untuk port TCP/UDP.

1. Buat dan inisialisasiIpPermissionmisalnya. GunakandenganIPv4Rangesmetode untuk mengatur berbagai alamat IP untuk mengotorisasi masuknya untuk, dan menggunakanwithIpProtocolmetode untuk mengatur protokol IP. GunakanwithFromPortdanwithToPortmetode untuk menentukan berbagai port untuk mengotorisasi masuknya untuk, 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 di`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 menginisialisasiAuthorizeSecurityGroupIngressRequestmisalnya. Gunakan`withGroupName`metode untuk menentukan nama grup keamanan, dan lulus`IpPermission`objek yang Anda inisialisasi sebelumnya kewithIpPermissionsmetode, sebagai berikut:

```
AuthorizeSecurityGroupIngressRequest authorizeSecurityGroupIngressRequest =
    new AuthorizeSecurityGroupIngressRequest();

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

3. Lulus objek permintaan ke dalamauthorizeSecurityGroupMasukmetode, sebagai berikut:

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

Jika Anda meneleponauthorizeSecurityGroupIngressdengan alamat IP yang masuknya sudah diotorisasi, metode ini melempar pengecualian. Membuat dan menginisialisasi baru`IpPermission`keberatan untuk mengotorisasi ingress untuk IP, port, dan protokol yang berbeda sebelum memanggil`AuthorizeSecurityGroupIngress`.

Setiap kali Anda menelepon`authorizeSecurityGroupMasuk`atau`authorizeSecurityGroupEgress`metode, 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, lihatAmazon EC2Pasangan kuncidi dalamAmazon EC2Panduan Pengguna untuk Instans Linux.

1. Membuat dan menginisialisasiCreateKeyPairRequestcontoh. MenggunakanwithKeyNamemetode 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 keCreateKeyPairmetode. Metode mengembalikanCreateKeyPairResultmisalnya, sebagai berikut:

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

3. Panggil objek hasilgetKeyPairmetode untuk mendapatkankeyPairobjek. MemanggilKeyPairobjekGetKeyMaterialmetode untuk mendapatkan terenkripsi kunci pribadi PEM-dikodekan, sebagai berikut:

```
KeyPair keyPair = new KeyPair();

keyPair = createKeyPairResult.getKeyPair();

String privateKey = keyPair.getKeyMaterial();
```

JalankanAmazon EC2Instans

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 menginisialisasiRunInstancesRequestcontoh. 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, lihatAmazon Machine Image (AMI).

withInstanceType

- Jenis instans yang kompatibel dengan AMI yang ditentukan. Untuk informasi selengkapnya, lihatTipe 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 atas`MinCount`. 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, lihatBuat Pasangan Kunci.

withSecurityGroups

- Satu atau lebih grup keamanan. Untuk informasi selengkapnya, lihatBuatAmazon EC2Grup Keamanan.

2. Luncurkan instance dengan melewatkan objek permintaan kerunInstancesmetode. Metode mengembalikanRunInstancesResultobjek, sebagai berikut:

```
RunInstancesResult result = amazonEC2Client.runInstances(
```

```
                runInstancesRequest);
```

Setelah instans Anda berjalan, Anda dapat terhubung dengannya menggunakan key pair Anda. Untuk informasi selengkapnya, lihatConnect ke Instans Linux Anda. diAmazon EC2Panduan Pengguna untuk Instans Linux.

Menggunakan Peran IAM untuk Memberikan Akses keAWSSumber Daya diAmazon EC2

Semua permintaan untukAmazon Web Services(AWS) harus ditandatangani secara kriptografis menggunakan kredensi yang dikeluarkan olehAWS. Anda dapat menggunakanIAM roleuntuk dengan mudah memberikan akses yang aman keAWSsumber daya dariAmazon EC2contoh.

Topik ini memberikan informasi tentang cara menggunakan peran IAM dengan aplikasi Java SDK berjalanAmazon EC2. Untuk informasi selengkapnya tentang instans IAM, lihatPeran IAM untukAmazon EC2di dalamAmazon EC2Panduan Pengguna untuk Instans Linux.

Rantai penyedia default dan profil instans EC2

Jika aplikasi Anda membuatAWSklien menggunakan konstruktor default, maka klien akan mencari kredensia menggunakanrantai 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. Kredensi disampaikan melaluiAmazon EC2layanan kontainer jika`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`variabel lingkungan diatur dan manajer keamanan memiliki izin untuk mengakses variabel.
5. Dikredensi profil instans, yang ada dalam metadata instance yang terkait dengan peran IAM untuk instans EC2.
6. Kredensi Token Identity Web dari lingkungan atau wadah.

Parameterkredensi profil instanslangkah dalam rantai penyedia default hanya tersedia saat menjalankan aplikasi Anda padaAmazon EC2misalnya, tetapi memberikan kemudahan terbesar penggunaan dan keamanan terbaik ketika bekerja denganAmazon EC2contoh. Anda juga dapat

melewati sebuahInstanceProfileCredentialsProvidercontoh 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 sementaraAWSkredensia yang memiliki izin yang sama dengan yang terkait dengan peran IAM yang terkait denganAmazon EC2misalnya dalam profil instans. Meskipun kredensi ini bersifat sementara dan akhirnya akan berakhir,`InstanceProfileCredentialsProvider`secara berkala menyegarkan mereka untuk Anda sehingga kredensi yang diperoleh terus memungkinkan akses keAWS.

> **Important**
>
> Penyegaran kredenal otomatis terjadicumaketika 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 kredensi profil instans, Anda bertanggung jawab untuk memeriksa dan menyegarkan kredensi kedaluwarsa.

Jika konstruktor klien tidak dapat menemukan kredensia menggunakan rantai penyedia kredensia, itu akan melemparAmazonClientException.

Panduan: Menggunakan peran IAM untuk instans EC2

Panduan berikut menunjukkan kepada Anda cara untuk mengambil objek dariAmazon S3menggunakan peran IAM untuk mengelola akses.

Buat IAM Role

Buat peran IAM yang memberikan akses hanya bacaAmazon S3.

1. Buka konsol IAM.
2. Di panel navigasi, pilihPeranmakaBuat Peran Baru.

3. Masukkan nama peran, lalu pilihLangkah Selanjutnya. Ingat nama ini, karena Anda akan membutuhkannya ketika Anda meluncurkanAmazon EC2contoh.
4. PadaPilih Tipe Peranhalaman, di bawah Layanan AWSPeran, pilih Amazon EC2 .
5. PadaMengatur Izinhalaman, di bawahPilih Templat Kebijakan, pilih Amazon S3Akses Hanya BacamakaLangkah Selanjutnya.
6. PadaTinjauhalaman, pilihBuat Peran.

Luncurkan Instans EC2 dan Tentukan Peran IAM Anda

Anda dapat meluncurkanAmazon EC2instans dengan peran IAM menggunakanAmazon EC2konsol atauAWS SDK for Java.

- Untuk meluncurkan sebuahAmazon EC2instans menggunakan konsol, ikuti petunjuk dalamMemulai denganAmazon EC2Instans Linuxdi dalamAmazon EC2Panduan Pengguna untuk Instans Linux.

  Ketika Anda mencapaiMeninjau Peluncuran instanshalaman, pilihMengedit detail instans. MasukPeran 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 sebuahAmazon EC2instans dengan peran IAM menggunakanAWS SDK for Java, lihatJalankan sebuahAmazon EC2Instans.

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, salinAWS SDK for Javaperpustakaan ke direktori yang baru dibuat. Jika Anda mengunduhAWS SDK for Javake`~/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 sajaGetS3Object.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 melemparAmazonServiceException. 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 keAmazon EC2contoh menggunakan salinan aman ( ), bersama denganAWS SDK for Javaperpustakaan. 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 Penggunamungkin "ec2-user", "root", atau "ubuntu". Untuk mendapatkan nama DNS publik dari instans Anda, bukaKonsol EC2dan carilahDNS Publikvalue dalamDeskripsitab (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
- sang`lib`dan`third-party`direktori adalah folder pustaka yang sesuai dariAWS SDK for Java.
- Parameter`-r`switch menunjukkan bahwa`scp`harus melakukan salinan rekursif dari semua isi`library`dan`third-party`direktori dalamAWS SDK for Javadistribusi.
- 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 keAmazon EC2contoh. Untuk informasi selengkapnya, lihatConnect ke Instans Linux Andadi dalamAmazon EC2Panduan 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 AndaAmazon S3keberatan jendela perintah Anda.

Tutorial:Amazon EC2 Instans Spot

Gambaran Umum

Instans Spot memungkinkan Anda untuk menawar pada tidak digunakanAmazon Elastic Compute Cloud(Amazon EC2) kapasitas hingga 90% dibandingkan harga Instans Pesanan dan menjalankan instans yang diperoleh selama tawaran Anda melebihi saat iniHarga Spot.Amazon EC2mengubah 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 menurunkanAmazon EC2biaya 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 EC2menyesuaikan 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 lainnyaAmazon EC2instance saat berjalan, dan seperti lainnyaAmazon EC2Instans 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 olehAmazon EC2, Anda tidak akan dikenakan biaya untuk sebagian jam penggunaan.

Tutorial ini menunjukkan cara menggunakanAWS SDK for Javauntuk melakukan hal berikut.

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

Prasyarat

Untuk menggunakan tutorial ini Anda harus memilikiAWS SDK for Javadiinstal, serta telah memenuhi prasyarat instalasi dasar. LihatSiapkan perangkatAWS SDK for Javauntuk informasi lebih lanjut.

Langkah 1: Menyiapkan Kredensi Anda

Untuk mulai menggunakan contoh kode ini, Anda perlu mengaturAWSkredensi. LihatMengaturAWSKredensial dan Wilayah untuk Pembangunanuntuk petunjuk tentang cara melakukannya.

> **Note**
>
> Kami menyarankan Anda menggunakan kredensi pengguna IAM untuk memberikan nilai-nilai ini. Untuk informasi selengkapnya, lihatMendaftar keAWSBuat Pengguna IAM.

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

Langkah 2: Menyiapkan Grup Keamanan

SEBUAHgrup keamananbertindak 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.createSecurityGroup`API 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 perangkatAWS Toolkit for Eclipse. LihatMengelola Grup Keamanan dariAWS Cost Exploreruntuk 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; kunjungiAmazon EC2Jenis 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. PilihLuncurkan Instanstombol.
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 lihatPenawaran untuk Instans Spotvideo. 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 PermintaanAnda 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 satuAWS Management ConsoleatauAmazon EC2API. Untuk informasi selengkapnya, kunjungiMelihat 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 HasilAnda 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 CepatAnda 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 membukaAmazon EC2Halaman harga. Saat Instance Spot berjalan, Anda membayar harga Spot yang berlaku untuk jangka waktu instans Anda berjalan. Harga Instans Spot ditetapkan olehAmazon EC2dan 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 menghubungi`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, kunjungiTutorial: LanjutanAmazon EC2Manajemen permintaan SpotatauRequestSpotInstanceskelas diAWS SDK for JavaReferensi 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 memilihdescribeSpotInstanceRequestsmetode untuk status ID permintaan Spot yang ingin kita pantau.

ID permintaan yang dibuat di Langkah 2 disematkan dalam menanggapi kami`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 tertunggakdanmengakhiri 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("Reponse 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`Requests`yang melakukan tindakan ini.

Kami juga membuat`GettingStarted`App kelas, yang memiliki metode utama di mana kita melakukan panggilan fungsi tingkat tinggi. Secara khusus, kami menginisialisasi`Requests`objek

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 diGitHub.

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

Langkah Selanjutnya

Lanjutkan denganTutorial: LanjutanAmazon EC2Manajemen permintaan Spot.

Tutorial: LanjutanAmazon EC2Manajemen permintaan spot

Amazon EC2Instans Spot memungkinkan Anda untuk menawar pada yang tidak digunakanAmazon EC2kapasitas dan menjalankan instans tersebut selama tawaran Anda melebihi saat iniharga spot.Amazon EC2mengubah harga spot secara berkala berdasarkan penawaran dan permintaan. Untuk informasi selengkapnya tentang Instans Spot Instans, lihatSpot Instancedi dalamAmazon EC2Panduan Pengguna untuk Instans Linux.

Prasyarat

Untuk menggunakan tutorial ini Anda harus memilikiAWS SDK for Javadiinstal, serta telah memenuhi prasyarat instalasi dasar. LihatMenyiapkanAWS SDK for JavaUntuk informasi lebih lanjut.

Menyiapkan kredenensi Anda

Untuk mulai menggunakan sampel kode ini, Anda perlu mengaturAWSkredensi. LihatMengaturAWSKredensi dan Wilayah untuk Pembangunanuntuk petunjuk tentang cara melakukannya.

> **Note**
>
> Sebaiknya Anda menggunakan kredenensiIAMpengguna untuk memberikan nilai-nilai ini. Untuk informasi selengkapnya, lihatMendaftarAWSdan BuatIAMPengguna.

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.createSecurityGroup`API 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.java`sampel kode. Perhatikan bahwa Anda hanya perlu menjalankan aplikasi ini sekali untuk membuat grup keamanan baru.

> **Note**
>
> Anda juga dapat membuat grup keamanan menggunakanAWS Toolkit for Eclipse. LihatMengelola Grup Keamanan dariAWS Cost Explorerdi dalamAWS Toolkit for EclipsePanduan Pengguna untuk informasi lebih lanjut.

Opsi pembuatan permintaan Instans Spot Instance

Seperti yang kita jelaskan diTutorial: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);
```

MengelompokkanAmazon EC2Permintaan 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 menentukangrup penempatanjika Anda menggunakan Instans Spot Komputasi Kinerja Tinggi (HPC), seperti instans komputasi klaster atau instans GPU klaster. 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.java`kelas.

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 (AmazonAmazon 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 terkaitAmazon Elastic Block Store(Amazon EBS) ke sebuahAmazon EBSobjek yang telah kita konfigurasi untuk`not`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 AmazonAmazon EBSvolume yang ingin Anda lampirkan ke Instans Spot setelah dilanjutkan. Anda melakukan ini hanya dengan menentukan ID snapshot di Anda`EbsBlockDevice`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 kunjungiMengelola Interupsivideo.

Cara menandai permintaan dan instans spot Anda

Menambahkan tanda keAmazon EC2sumber 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 penandaanAmazon EC2sumber daya, pergi keMenggunakan Tagdi dalamAmazon EC2Panduan Pengguna untuk Instans Linux.

Permintaan penandaan

Untuk menambahkan tag ke permintaan tempat Anda, Anda perlu memberi tag pada merekasetelahmereka telah diminta. Nilai kembali dari`requestSpotInstances()`menyediakan Anda denganRequestSpotInstancesResultobjek 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 keCreateTagsRequestdan memanggilAmazon EC2klien`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 menghubungiAmazon EC2kliendescribeSpotInstanceRequests()metode dengandescribeSpotInstanceRequestsRequestobjek. KembaliandescribeSpotInstanceRequestsResultobjek berisi daftarSpotInstanceRequestobjek 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 dariSpotInstanceRequestobjek dengan memanggil nyagetInstanceId()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 Instans`cancelSpotInstanceRequests`padaAmazon 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 EC2klien`terminateInstances()`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`Requests`yang melakukan tindakan ini. Kami juga membuat`GettingStarted`App kelas, yang memiliki metode utama di mana kita melakukan panggilan fungsi tingkat tinggi.

Kode sumber lengkap untuk contoh ini dapat dilihat atau diunduh diGitHub.

Selamat! Anda telah menyelesaikan tutorial Fitur Permintaan Lanjutan untuk mengembangkan perangkat lunak Instans Spot denganAWS SDK for Java.

MengelolaAmazon EC2Instans

Membuat Instans

MembuatAmazon EC2contoh dengan memanggil Amazonec2Client`runInstances`metode, menyediakannya denganRunInstancesRequestmengandungAmazon Machine Image (AMI)untuk menggunakan dantipe 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();
```

LihatLengkapi Contoh.

Memulai Instans

Untuk memulaiAmazon EC2misalnya, 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);
```

LihatLengkapi Contoh.

Menghentikan Instans

Untuk menghentikanAmazon EC2misalnya, 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);
```

LihatLengkapi Contoh.

Mem-boot Ulang Instans

Untuk me-rebootAmazon EC2misalnya, hubungi Amazonec2Client`rebootInstances`metode, menyediakannya denganRebootInstancesRequestberisi 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);
```

LihatLengkapi Contoh.

Menjelaskan Instans

Untuk mencantumkan instans Anda, buatDescribeInstancesRequestdan memanggil Amazonec2Client`describeInstances`metode. Ini akan mengembalikanDescribeInstancesResultobjek yang dapat Anda gunakan untuk daftarAmazon EC2instans untuk akun dan wilayah Anda.

Instans dikelompokkan berdasarkankekhawatiran. Setiap reservasi sesuai dengan panggilan k`estartInstances`yang meluncurkan instance. Untuk mencantumkan instans Anda, Anda harus memanggil`DescribeInstancesResult`kelas`getReservations' method, and then call` `getInstances`pada setiap kembaliReservasiobjek.

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 hasil`getNextToken`metode untuk objek permintaan asli Anda`setNextToken`metode,

kemudian menggunakan objek permintaan yang sama dalam panggilan berikutnya untuk`describeInstances`.

LihatLengkapi Contoh.

Memantau Instans

Anda dapat memantau berbagai aspekAmazon EC2contoh, seperti CPU dan pemanfaatan jaringan, memori yang tersedia, dan ruang disk yang tersisa. Untuk mempelajari tentang pemantauan instans, lihatPemantauanAmazon EC2diAmazon EC2Panduan Pengguna Instans Linux.

Untuk memulai pemantauan sebuah instans, Anda harus membuatMonitorInstancesRequestdengan 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);
```

LihatLengkapi Contoh.

Memantau Instans

Untuk menghentikan pemantauan instance, buatUnmonitorInstancesRequestdengan ID instance untuk menghentikan pemantauan, dan menyebarkannya ke Amazonec2Clientun`monitorInstances`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);
```

LihatLengkapi Contoh.

Informasi Selengkapnya

- RunInstancesdiAmazon EC2Referensi API
- DescribeInstancesdiAmazon EC2Referensi API
- StartInstancesdiAmazon EC2Referensi API
- StopInstancesdiAmazon EC2Referensi API
- RebootInstancesdiAmazon EC2Referensi API
- MonitorInstancesdiAmazon EC2Referensi API
- UnmonitorInstancesdiAmazon EC2Referensi API

Menggunakan Alamat Elastic IPAmazon EC2

EC2-Classic pensiun

> **Warning**
>
> Kami pensiun EC2-Classic pada 15 Agustus 2022. Kami menyarankan Anda untuk memindahkan dari EC2-Classic ke VPC. Untuk informasi selengkapnya, lihatMemindahkan dari EC2-Classic ke VPCdi dalamPanduan Pengguna Amazon EC2 untuk Instans LinuxatauPanduan Pengguna Amazon EC2 untuk Instans Windows. Lihat juga posting blogEC2-Classic-Classic Networking Pensiun - Inilah Cara Mempersiapkan.

Mengalokasikan Alamat Elastic IP

Untuk menggunakan alamat Elastic IP, pertama-tama Anda mengalokasikannya ke akun Anda, lalu mengaitkannya dengan instans atau antarmuka jaringan.

Untuk mengalokasikan alamat IP Elastis, hubungi Amazonec2Client`allocateAddress`Metode dengan`AllocateAddressRequest`objek yang berisi jenis jaringan (EC2 klasik atau VPC).

Yang dikembalikan`AllocateAddressResult`berisi ID alokasi yang dapat Anda gunakan untuk mengasosiasikan alamat dengan instance, dengan meneruskan ID alokasi dan ID instance dalam`AssociateAddressRequest`ke Amazonec2Client`associateAddress`Metode.

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);
```

LihatLengkapi Contoh.

Menggambarkan Alamat Elastic IP

Untuk mencantumkan alamat IP Elastic yang ditetapkan ke akun Anda, hubungi Amazonec2Client's`describeAddresses`Metode. Ini menghasilkanDescribeAddressesResultyang dapat Anda gunakan untuk mendapatkan daftarAlamatobjek yang mewakili alamat IP Elastic pada 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());
}
```

LihatLengkapi Contoh.

Melepas alamat Elastic IP

Untuk merilis alamat IP Elastis, hubungi Amazonec2Client`releaseAddress`metode, melewatinyaReleaseAddressRequestberisi ID alokasi alamat IP Elastis yang ingin Anda rilis.

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, alamat IPAWSPangkalan alamat IP dan mungkin tidak tersedia untuk Anda sesudahnya. Pastikan untuk memperbarui data DNS Anda dan server atau perangkat apa pun yang berkomunikasi dengan alamat tersebut. Jika Anda mencoba merilis alamat IP Elastis yang Anda rilis, Anda akan mendapatkanAuthFailurekesalahan jika alamat sudah dialokasikan ke yang lainAkun AWS.

Jika Anda menggunakanEC2-ClassicatauVPC default, kemudian melepaskan alamat IP Elastic secara otomatis memisahkannya dari instance apa pun yang terkait dengannya. Untuk memisahkan alamat IP Elastic tanpa melepaskannya, gunakan Amazonec2Client`disassociateAddress`Metode.

Jika Anda menggunakan VPC non-default, Andaharusmenggunakan`disassociateAddress`untuk memisahkan alamat IP Elastis sebelum Anda mencoba melepaskannya. Jika tidak,Amazon EC2menghasilkan kesalahan (InvalidIPAddress.InUse).

LihatLengkapi Contoh.

Informasi Selengkapnya

- Alamat Elastic IPdi dalamAmazon EC2Panduan Pengguna untuk Instans Linux
- AllocateAddressdi dalamAmazon EC2Referensi API
- DescribeAddressesdi dalamAmazon EC2Referensi API
- ReleaseAddressdi dalamAmazon EC2Referensi API

Gunakan wilayah dan availability zone

Mendeskripsikan wilayah

Untuk mencantumkan Wilayah yang tersedia untuk akun Anda, hubungi Amazonec2Client`describeRegions`metode. Ini menghasilkanDescribeRegionsResult. Panggil objek yang dikembalikan`getRegions`metode untuk mendapatkan daftarWilayahbenda-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());
}
```

LihatLengkapi Contoh.

Mendeskripsikan zona ketersediaan

Untuk mencantumkan setiap Availability Zone yang tersedia untuk akun Anda, hubungi Amazonec2Client`describeAvailabilityZones`metode. Ini menghasilkanDescribeAvailabilityZonesResult. Memanggil`getAvailabilityZones`metode untuk mendapatkan daftarAvailabilityZoneobjek 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());
}
```

LihatLengkapi Contoh.

Mendeskripsikan akun

Untuk menggambarkan akun Anda, hubungi Amazonec2Client`describeAccountAttributes`metode. Metode ini mengembalikanDescribeAccountAttributesResultobjek. Memanggil objek ini`getAccountAttributes`metode untuk mendapatkan daftarAccountAttributeobjek. Anda dapat iterate melalui daftar untuk mengambilAccountAttributeobjek.

Anda bisa mendapatkan nilai atribut akun Anda dengan menerapkanAccountAttributeobjek`getAttributeValues`metode. Metode ini menghasilkan daftarAccountAttributeValueobjek. 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.getStackTrace();
}
```

LihatLengkapi Contohdi GitHub.

Informasi lain

- Wilayah dan Availability Zonedi dalamAmazon EC2Panduan Pengguna untuk Instans Linux
- DescribeRegionsdi dalamAmazon EC2Referensi API
- DescribeAvailabilityZonesdi dalamAmazon EC2Referensi API

Bekerja denganAmazon EC2Pasangan 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());
}
```

LihatLengkapi Contoh.

Menghapus Pasangan Kunci

Untuk menghapus key pair, panggil Amazonec2Client`deleteKeyPair`metode, melewatinyaDeleteKeyPairRequestyang 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);
```

LihatLengkapi Contoh.

Informasi Selengkapnya

- Amazon EC2Pasangan KuncidiAmazon EC2Panduan Pengguna untuk Instans Linux
- CreateKeyPairdiAmazon EC2Referensi API

- DescribeKeyPairsdiAmazon EC2Referensi API
- DeleteKeyPairdiAmazon EC2Referensi API

Bekerja dengan GrupAmazon EC2

Membuat Grup Keamanan

Untuk membuat grup keamanan, panggil`createSecurityGroup` metode Amazonec2Client dengan CreateSecurityGroupRequestyang 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) keAmazon 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 AuthorizeSecurityGroupIngressRequestobjek. 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 AuthorizeSecurityGroupEgressRequestke`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 DescribeSecurityGroupsResultyang dapat Anda gunakan untuk mengakses daftar kelompok keamanan dengan memanggil`getSecurityGroups` metodenya, yang mengembalikan daftar SecurityGroupobjek.

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 DeleteSecurityGroupRequestyang 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
- CreateSecurityGroupdi ReferensiAmazon EC2 API
- DescribeSecurityGroupsdi ReferensiAmazon EC2 API
- DeleteSecurityGroupdi ReferensiAmazon EC2 API
- AuthorizeSecurityGroupIngressdi ReferensiAmazon EC2 API

Contoh IAM MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrogramanIAMdengan menggunakanAWS SDK for Java.

AWS Identity and Access Management(IAM) memungkinkan Anda untuk mengontrol akses keAWSlayanan 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, kunjungiIAMPanduan Pengguna.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parametercontoh 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 Kunci Akses

Untuk membuat kunci akses IAM, panggil AmazonIdentityManagementClient`createAccessKey`Metode 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`createAccessKey`Metode.

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, buatListAccessKeysRequestobjek yang berisi nama pengguna untuk daftar kunci untuk, dan meneruskannya ke AmazonIdentityManagementClient`listAccessKeys`Metode.

> **Note**
>
> Jika Anda tidak memberikan nama pengguna ke`listAccessKeys`, ia akan mencoba untuk daftar kunci akses yang terkait denganAkun AWSyang 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 dikembalikanListAccessKeysResultobjek 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`.

LihatLengkapi Contohdi 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 menggunakanGetAccessKeyLastUsedRequestobjek, atau langsung ke overload yang mengambil kunci akses ID langsung.

Anda kemudian dapat menggunakan kembaliGetAccessKeyLastUsedResultobjek 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());
```

LihatLengkapi Contohdi GitHub.

Mengaktifkan atau Menonaktifkan Kunci Akses

Anda dapat mengaktifkan atau menonaktifkan kunci akses dengan membuatUpdateAccessKeyRequestobjek, menyediakan ID kunci akses, opsional nama pengguna, dan yang diinginkanStatus, kemudian meneruskan objek permintaan ke AmazonIdentityManagementClient's`updateAccessKey`Metode.

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);
```

LihatLengkapi Contohdi GitHub.

Menghapus Access key

Untuk menghapus kunci akses secara permanen, panggil AmazonIdentityManagementClient's`deleteKey`metode, menyediakannya dengandeleteAccessKeyRequestberisi 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, gunakanupdateAccessKeyMetode 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);
```

LihatLengkapi Contohdi GitHub.

Informasi Selengkapnya

- CreateAccessKeydalam Referensi API IAM
- ListAccessKeysdalam Referensi API IAM
- GetAccessKeyLastUseddalam Referensi API IAM
- UpdateAccessKeydalam Referensi API IAM
- DeleteAccessKeydalam Referensi API IAM

Membuat pengguna IAM

Membuat pengguna

Buat pengguna IAM baru dengan memberikan nama pengguna untuk`createUser` metodeAmazonIdentityManagementClient ini, baik secara langsung atau menggunakan CreateUserRequestobjek 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 lengkapnyaGitHub.

Daftar pengguna

Untuk mencantumkan pengguna IAM untuk akun Anda, buat yang baru ListUsersRequestdan teruskan ke`listUsers` metode tersebutAmazonIdentityManagementClient. Anda dapat mengambil daftar pengguna dengan memanggil`getUsers` ListUsersResultobjek yang dikembalikan.

Daftar pengguna yang dikembalikan`listUsers` oleh paged. Anda dapat memeriksa untuk melihat ada lebih banyak hasil untuk diambil dengan memanggil`getIsTruncated` 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 lengkapnyaGitHub.

Membuat pengguna

Untuk memperbarui pengguna, panggil`updateUser` metodeAmazonIdentityManagementClient objek, yang mengambil UpdateUserRequestobjek 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 lengkapnyaGitHub.

Membuat pengguna

Untuk menghapus pengguna, panggilAmazonIdentityManagementClient`deleteUser` permintaan dengan UpdateUserRequestobjek 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 lengkapnyaGitHub.

Informasi Selengkapnya

- Pengguna IAM dalam PanduanIAM Pengguna
- Mengelola Pengguna IAM di PanduanIAM Pengguna
- CreateUserdalam Referensi IAM API
- ListUsersdalam Referensi IAM API
- UpdateUserdalam Referensi IAM API

- DeleteUserdalam 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**
>
> AWSmendukung tepat satu alias akun per akun.

Membuat Alias akun

Untuk membuat alias akun, panggil`createAccountAlias` metode dengan CreateAccountAliasRequestobjek 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 lengkapnyaGitHub.

Alias akun

Untuk mencantumkan alias akun Anda, jika ada, panggil`listAccountAliases` metode tersebutAmazonIdentityManagementClient.

> **Note**
>
> Yang dikembalikan ListAccountAliasesResultmendukung metode yang samagetIsTruncated dan sepertigetMarker metodeAWS SDK for Java daftar lainnya, tetapi hanyaAkun 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 lengkapnyaGitHub.

Menghapus.

Untuk menghapus alias akun Anda, panggildeleteAccountAlias metode tersebutAmazonIdentityManagementClient. Saat menghapus alias akun, Anda harus memberikan namanya menggunakan DeleteAccountAliasRequestobjek.

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 lengkapnyaGitHub.

Informasi Selengkapnya

- IDAWS Akun Anda dan Aliasnya di PanduanIAM Pengguna
- CreateAccountAliasdalam Referensi IAM API
- ListAccountAliasesdalam Referensi IAM API
- DeleteAccountAliasdalam Referensi IAM API

Cara menggunakan kebijakan IAM

Membuat Kebijakan

Untuk membuat kebijakan baru, berikan nama kebijakan dan dokumen kebijakan yang diformat JSON dalamCreatePolicyRequestke 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 dengansintaks yang terdokumentasi dengan baik. Berikut ini adalah contoh yang menyediakan akses untuk membuat permintaan tertentuDynamoDB.

```
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\"" +
    "    }" +
    "   ]" +
    "}";
```

LihatLengkapi Contohdi GitHub.

Mendapatkan kebijakan

Untuk mengambil kebijakan yang ada, hubungi AmazonIdentityManagementClient's`getPolicy`metode, menyediakan ARN kebijakan dalamGetPolicyRequestobjek.

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);
```

LihatLengkapi Contohdi 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 dalamAttachRolePolicyRequest.

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);
```

LihatLengkapi Contohdi 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());
}
```

LihatLengkapi Contohdi GitHub.

Melepaskan kebijakan peran

Untuk melepaskan kebijakan dari peran, panggil AmazonIdentityManagementClient's`detachRolePolicy`metode, menyediakannya dengan nama peran dan kebijakan ARN dalamDetachRolePolicyRequest.

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);
```

LihatLengkapi Contohdi GitHub.

Informasi Selengkapnya

- Gambaran umum dari kebijakan IAMdiIAMPanduan Pengguna.

- AWSReferensi kebijakan IAMdiIAMPanduan Pengguna.
- CreatePolicydalam Referensi API IAM
- GetPolicydalam Referensi API IAM
- AttachRolePolicydalam Referensi API IAM
- ListAttachedRolePoliciesdalam Referensi API IAM
- DetachRolePolicydalam Referensi API IAM

Cara menggunakan Sertifikat Server IAM

Untuk mengaktifkan koneksi HTTPS ke situs web atau aplikasi AndaAWS, Anda membutuhkan SSL/TLSsertifikat server. Anda dapat menggunakan sertifikat server yang disediakan olehAWSCertificate 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 AndaAWSsumber daya, dan membiarkan ACM menangani pembaruan sertifikat untuk Anda. Sertifikat yang disediakan oleh ACM gratis. Untuk informasi selengkapnya tentang ACM, lihatPanduan Pengguna ACM.

Mendapatkan Sertifikat Server

Anda dapat mengambil sertifikat server dengan memanggil AmazonIdentityManagementClient's`getServerCertificate`metode, melewatinyaGetServerCertificateRequestdengan 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);
```

LihatLengkapi Contohdi GitHub.

Sertifikat Server

Untuk mencantumkan sertifikat server Anda, hubungi AmazonIdentityManagementClient'slistServerCertificatesmetode denganListServerCertificatesRequest. Ini menghasilkanListServerCertificatesResult.

Panggil yang dikembalikanListServerCertificateResultobjekgetServerCertificateMetadataListmetode untuk mendapatkan daftarServerCertificateMetadataobjek yang dapat Anda gunakan untuk mendapatkan informasi tentang setiap sertifikat.

Hasil dapat dipotong; jikaListServerCertificateResultobjekgetIsTruncatedpengembalian metodetrue, hubungiListServerCertificatesRequestobjeksetMarkermetode dan menggunakannya untuk memanggillistServerCertificateslagi 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;
    }
}
```

LihatLengkapi Contohdi GitHub.

Memperbarui Sertifikat Server

Anda dapat memperbarui nama atau jalur sertifikat server dengan memanggil AmazonIdentityManagementClient's`updateServerCertificate`metode. DibutuhkanUpdateServerCertificateRequestobjek 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);
```

LihatLengkapi Contohdi GitHub.

Menghapus Sertifikat Server

Untuk menghapus sertifikat server, hubungi AmazonIdentityManagementClient's`deleteServerCertificate`metode denganDeleteServerCertificateRequestberisi 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);
```

LihatLengkapi Contohdi GitHub.

Informasi Selengkapnya

- Bekerja dengan Sertifikat ServerdiIAMPanduan Pengguna
- GetServerCertificatedalam Referensi API IAM
- ListServerCertificatesdalam Referensi API IAM
- UpdateServerCertificatedalam Referensi API IAM
- DeleteServerCertificatedalam 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. Parameterkode 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

Bagian ini menyediakan contoh pemrograman denganLambdalayanan klien dengan menggunakanAWS SDK for Java. Untuk mempelajari cara membuatLambdafungsi, lihatCara MembuatAWS Lambdafungsi.

Topik

Mengaktifkan fungsi

Anda dapat memanggilLambdaberfungsi dengan membuatAWSLambdaobjek dan memohon`invoke`metode. BuatInvokeRequestobjek 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, panggilInvokeRequestobjek`withPayload`Metode 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 memanggilLambdafungsi.

```
        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 lengkapGitHub.

Fungsi daftar

MembangunAWSLambdaobjek dan memohon`listFunctions`metode. Metode ini mengembalikanListFunctionsResultobjek. Anda dapat memanggil objek ini`getFunctions`metode untuk mengembalikan daftarFunctionConfigurationobjek. 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 daftarLambdanama 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 lengkapGitHub.

Menghapus fungsi

MembangunAWSLambdaobjek dan memohon`deleteFunction`metode. BuatDeleteFunctionRequestobjek dan menyebarkannya ke`deleteFunction`metode. Objek ini berisi informasi seperti nama fungsi untuk menghapus. Nama fungsi muncul sebagaiarn:aws:lambda:us-east- 1:5556330391:fungsi: helloFunction. Anda dapat mengambil nilai dengan melihat fungsi diAWS 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 menghapusLambdafungsi.

```
        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 lengkapGitHub.

Amazon PinpointContoh MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrogramanAmazon Pinpointmenggunakan AWS SDK for Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parametercontoh 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 AplikasiAmazon Pinpoint
- Membuat Endpoint diAmazon Pinpoint
- Membuat Segmen diAmazon Pinpoint
- Membuat Kampanye diAmazon Pinpoint
- Saluran diAmazon Pinpoint

Membuat dan Menghapus AplikasiAmazon Pinpoint

Aplikasi adalahAmazon Pinpointproyek 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 diAmazon Pinpointdengan memberikan nama aplikasi keCreateAppRequestobjek, 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);
```

LihatLengkapi Contohdi GitHub.

Menghapus Aplikasi

Untuk menghapus aplikasi, hubungi AmazonPinpointClient`deleteApp`request 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);
```

LihatLengkapi Contohdi GitHub.

Informasi Selengkapnya

- Aplikasidi dalamAmazon PinpointReferensi API
- Aplikasidi dalamAmazon PinpointReferensi 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 dalamEndpointRequestobjek.

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 membuatUpdateEndpointRequestobjek 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());
```

Lihatcontoh lengkapdi GitHub.

Informasi Selengkapnya

- Menambahkan EndpointdiAmazon PinpointPanduan Pengembang
- Titik akhirdiAmazon PinpointReferensi API

Membuat Segmen diAmazon 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 diAmazon Pinpointdengan mendefinisikan dimensi segmen dalamSegmentDimensionsobjek.

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 mengaturSegmentDimensionsobjek dalamWriteSegmentRequest, yang pada gilirannya digunakan untuk membuatCreateSegmentRequestobjek. 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);
```

Lihatcontoh lengkapdi GitHub.

Informasi Selengkapnya

- Amazon PinpointSegmendiAmazon PinpointPanduan Pengguna
- Membuat SegmendiAmazon PinpointPanduan Pengembang
- SegmendiAmazon PinpointReferensi API

- SegmentdiAmazon PinpointReferensi API

Membuat Kampanye diAmazon 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 WriteCampaignRequestobjek.

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 baruAmazon Pinpoint WriteCampaignRequestdengan menyediakan konfigurasi kampanye ke CreateCampaignRequestobjek. Akhirnya, lewatiCreateCampaignRequest objek keAmazonPinpointClient`createCampaign` metode.

Kode

```
CreateCampaignRequest createCampaignRequest = new CreateCampaignRequest()
        .withApplicationId(appId).withWriteCampaignRequest(request);

CreateCampaignResult result = client.createCampaign(createCampaignRequest);
```

Lihat contoh lengkapnyaGitHub.

Informasi Selengkapnya

- Amazon PinpointKampanye di PanduanAmazon Pinpoint Pengguna
- Membuat Kampanye di PanduanAmazon Pinpoint Pengembang
- Kampanye di ReferensiAmazon Pinpoint API
- Kampanye di ReferensiAmazon Pinpoint API
- Aktivitas Kampanye di ReferensiAmazon Pinpoint API
- Versi Kampanye di ReferensiAmazon Pinpoint API
- Versi Kampanye di ReferensiAmazon Pinpoint API

Saluran diAmazon Pinpoint

Saluran mendefinisikan jenis platform tempat Anda dapat mengirimkan pesan. Contoh ini menunjukkan cara menggunakan saluran APN untuk mengirim pesan.

Perbarui Saluran

Aktifkan saluran diAmazon Pinpointdengan menyediakan ID aplikasi dan objek permintaan dari jenis saluran yang ingin Anda perbarui. Contoh ini memperbarui saluran APN, yang

memerlukanAPNSChannelRequestobjek. Set ini diUpdateApnsChannelRequestdan 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);
```

Lihatcontoh lengkapdi GitHub.

Informasi Selengkapnya

- Amazon PinpointChanneldi dalamAmazon PinpointPanduan Pengguna
- Saluran ADMdi dalamAmazon PinpointReferensi API
- Salurandi dalamAmazon PinpointReferensi API
- Saluran APNdi dalamAmazon PinpointReferensi API
- Saluran APNdi dalamAmazon PinpointReferensi API
- Saluran Sandbox VoIP APNdi dalamAmazon PinpointReferensi API
- Salurandi dalamAmazon PinpointReferensi API
- Salurandi dalamAmazon PinpointReferensi API
- Saluran GCMdi dalamAmazon PinpointReferensi API
- Saluran SMSdi dalamAmazon PinpointReferensi API

Amazon S3Contoh MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrogramanAmazon S3menggunakanAWS SDK for Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parametercontoh 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 MenghapusAmazon S3Bucket

Setiap objek (file) diAmazon S3harus berada dalamember, yang merupakan koleksi (wadah) objek. Setiap ember dikenal olehkunci(nama), yang harus unik. Untuk informasi mendetail tentang bucket dan konfigurasinya, lihatBekerja denganAmazon S3Bucketdi dalamAmazon Simple Storage ServicePanduan Pengguna.

> **Note**
>
> Praktik terbaik
> Kami sarankan Anda mengaktifkanAbortIncompleteMultipartUploadaturan siklus hidup diAmazon S3ember.
> Aturan ini mengarahkanAmazon S3untuk membatalkan unggahan multi-bagian yang tidak selesai dalam jumlah hari tertentu setelah dimulai. Ketika batas waktu yang ditetapkan

terlampaui,Amazon S3membatalkan upload dan kemudian menghapus data upload yang tidak lengkap.
Untuk informasi selengkapnya, lihatKonfigurasi Siklus Hidup untuk Bucket dengan Versioningdi dalamAmazon S3Panduan Pengguna.

> **Note**
>
> Contoh kode ini berasumsi bahwa Anda memahami materi diMenggunakanAWS SDK for Javadan telah dikonfigurasi defaultAWSkredensi menggunakan informasi diMengaturAWSKredensial dan Wilayah untuk Pembangunan.

Buat Bucket

Gunakan klien AmazonS3`createBucket`metode. Yang baruBucketdikembalikan. 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;
```

LihatLengkapi Contohdi GitHub.

Buat Daftar Bucket

Gunakan klien Amazon S3`listBucket`metode. Jika berhasil, daftarBucketdikembalikan.

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());
}
```

LihatLengkapi Contohdi GitHub.

Hapus Bucket

Sebelum Anda dapat menghapusAmazon S3ember, Anda harus memastikan bahwa ember kosong atau kesalahan akan menghasilkan. Jika Anda memilikiember berversi, Anda juga harus menghapus objek berversi yang terkait dengan bucket.

> **Note**
>
> ParameterLengkapi Contohmencakup masing-masing langkah ini secara berurutan, memberikan solusi lengkap untuk menghapusAmazon S3ember 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 Amazon`S3`metode untuk mengambil daftar objek dan`listObjects`untuk menghapus masing-masing.`deleteObject`

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;
    }
}
```

LihatLengkapi Contohdi GitHub.

Hapus Objek dari Bucket Berversi Sebelum Menghapusnya

Jika Anda menggunakanember 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 Amazon`S3listVersions`metode untuk daftar objek berversi, dan kemudian`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;
    }
}
```

LihatLengkapi Contohdi GitHub.

Hapus Bucket Kosong

Setelah Anda menghapus objek dari ember (termasuk objek berversi), Anda dapat menghapus bucket itu sendiri dengan menggunakan klien AmazonS3`deleteBucket`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);
```

LihatLengkapi Contohdi 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 kredensyal default menggunakan informasi di Menyiapkan AWS Kredensyal 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 eror 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.getErrorMessage());
    System.exit(1);
}
```

Lihat contoh lengkapnyaGitHub.

Daftar Objek

Untuk mendapatkan daftar objek dalam bucket, gunakan `listObjects` metode klien AmazonS3, yang menyediakan nama bucket.

`listObjects`Metode mengembalikan sebuah ObjectListingobjek 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 lengkapnyaGitHub.

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 eror 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 lengkapnyaGitHub.

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.getErrorMessage());
    System.exit(1);
}
System.out.println("Done!");
```

Lihat contoh lengkapnyaGitHub.

> **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.getErrorMessage());
    System.exit(1);
```

```
}
```

Lihat contoh lengkapnyaGitHub.

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 lengkapnyaGitHub.

MengelolaAmazon S3Akses Izin untuk Bucket dan Objek

Anda dapat menggunakan daftar kontrol akses (ACL) untukAmazon S3ember dan objek untuk kontrol halus atas AndaAmazon S3sumber daya.

> **Note**
>
> Contoh-contoh kode ini berasumsi bahwa Anda memahami materi diMenggunakanAWS SDK for Javadan telah dikonfigurasi defaultAWSkredensi menggunakan informasi diMengaturAWSKredensial dan Wilayah untuk Pembangunan.

Dapatkan Daftar Kontrol Akses untuk Bucket

Untuk mendapatkan ACL saat ini untuk ember, hubungi AmazonS3`getBucketAcl`metode, melewatinyanama bucketkueri. Metode ini mengembalikanAccessControlListobjek. Untuk mendapatkan setiap hibah akses dalam daftar, hubungi`getGrantsAsList`metode, yang akan mengembalikan daftar Java standarPemberian Izinbenda.

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);
}
```

LihatLengkapi Contohdi GitHub.

Mengatur Access Control List untuk Bucket

Untuk menambah atau memodifikasi izin ke ACL untuk bucket, hubungi AmazonS3`setBucketAcl`metode. DibutuhkanAccessControlListobjek 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.getErrorMessage());
    System.exit(1);
}
```

> **Note**
>
> Anda dapat memberikan pengenal unik penerima hibah secara langsung menggunakanPenerima izinkelas, atau menggunakanEmailAddressGranteekelas untuk mengatur penerima hibah melalui email, seperti yang telah kita lakukan di sini.

LihatLengkapi Contohdi GitHub.

Dapatkan Daftar Kontrol Akses untuk Objek

Untuk mendapatkan ACL saat ini untuk sebuah objek, panggil AmazonS3`getObjectAcl`metode, melewatinyanama bucketdannama objekkueri. LIKE`getBucketAcl`, metode ini mengembalikanAccessControlListobjek yang dapat Anda gunakan untuk memeriksa masing-masingPemberian 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);
}
```

LihatLengkapi Contohdi GitHub.

Mengatur Access Control List untuk Object

Untuk menambah atau memodifikasi izin ke ACL untuk objek, panggil AmazonS3`setObjectAcl`metode. DibutuhkanAccessControlListobjek 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 menggunakanPenerima izinkelas, atau menggunakanEmailAddressGranteekelas untuk mengatur penerima hibah melalui email, seperti yang telah kita lakukan di sini.

LihatLengkapi Contohdi GitHub.

Informasi Selengkapnya

- acl DAPATKAN emberdiAmazon S3Referensi API
- acl LETAKKAN emberdiAmazon S3Referensi API
- acl DAPATKAN ObjekdiAmazon S3Referensi API
- acl LETAKKAN ObjekdiAmazon S3Referensi API

Mengelola akses keAmazon S3Bucket Menggunakan Kebijakan ember

Anda dapat mengatur, mendapatkan, atau menghapuskebijakan bucketuntuk mengelola akses keAmazon S3ember.

Mengatur Kebijakan ember

Anda dapat mengatur kebijakan bucket S3 tertentu dengan:

- Memanggil klien Amazon`S3setBucketPolicy`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.getErrorMessage());
    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 menggunakanKebijakankelas

Dengan menggunakan`Policy`kelas, Anda tidak perlu khawatir tentang benar memformat string teks Anda. Untuk mendapatkan teks kebijakan JSON dari`Policy`kelas, menggunakan nya`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();
```

ParameterPolicykelas juga menyediakanfromJsonmetode 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 denganIllegalArgumentExceptionjika 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.

LihatLengkapi Contohdi GitHub.

Dapatkan Kebijakan ember

Untuk mengambil kebijakan untukAmazon S3ember, hubungi klien AmazonS3getBucketPolicymetode, 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 dalamAmazon Simple Storage ServicePanduan Pengguna

- Contoh Kebijakan emberdi dalamAmazon Simple Storage ServicePanduan Pengguna

Menggunakan TransferManager untukAmazon S3Operasi

Anda dapat menggunakanAWS SDK for Java TransferManager kelas untuk andal mentransfer file dari lingkungan lokal keAmazon S3dan 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 mengaktifkanAbortIncompleteMultipartUploadaturan siklus hidupAmazon S3ember.
> Aturan ini mengarahkanAmazon S3untuk membatalkan unggahan multi-bagian yang tidak selesai dalam jumlah hari tertentu setelah dimulai. Ketika batas waktu yang ditetapkan terlampaui,Amazon S3membatalkan upload dan kemudian menghapus data upload yang tidak lengkap.
> Untuk informasi selengkapnya, lihatKonfigurasi Siklus Hidup untuk Bucket dengan VersioningdiAmazon S3Panduan Pengguna.

> **Note**
>
> Contoh-contoh kode ini berasumsi bahwa Anda memahami materi diMenggunakanAWS SDK for Javadan telah dikonfigurasi defaultAWSkredensi menggunakan informasi diMengaturAWSKredensial dan Wilayah untuk Pembangunan.

Upload File dan Direktori

TransferManager dapat meng-upload file, daftar file, dan direktori ke setiapAmazon S3ember yang Anda milikisebelumnya dibuat.

Topik

- Meng-unggah File Tunggal
- Unggah Daftar File

- Meng-unggah sebuah Direktori

Meng-unggah File Tunggal

Hubungi TransferManager`upload`metode, menyediakanAmazon S3nama bucket, kunci (objek) nama, dan Java standarBerkasobjek 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.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Parameter`upload`metode pengembalianlangsung, menyediakan`Upload`objek yang akan digunakan untuk memeriksa status transfer atau menunggu sampai selesai.

LihatTunggu Transfer untuk Menyelesaikanuntuk 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. LihatDapatkan Status Transfer dan Kemajuanuntuk informasi lebih lanjut.

LihatLengkapi Contohdi GitHub.

Unggah Daftar File

Untuk mengunggah beberapa file dalam satu operasi, panggil TransferManager`uploadFileList`, memberikan hal berikut:

- SesiAmazon S3nama bucket
- SEBUAHprefiks kunciuntuk menambahkan nama-nama objek yang dibuat (jalan dalam ember di mana untuk menempatkan objek)
- SEBUAHBerkasobjek yang mewakili direktori relatif dari yang untuk membuat path file
- SEBUAHDaftarobjek yang berisi satu setBerkasobjek 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.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

LihatTunggu Transfer untuk Menyelesaikanuntuk 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. LihatDapatkan Status Transfer dan Kemajuanuntuk informasi lebih lanjut.

ParameterMultipleFileUploadobjek yang dikembalikan oleh`uploadFileList`dapat digunakan untuk query negara transfer atau kemajuan. LihatPoll Kemajuan Transfer Saat InidanDapatkan Transfer Progress dengan ProgressListeneruntuk informasi lebih lanjut.

Anda juga dapat menggunakan`MultipleFileUpload`'s`getSubTransfers`metode untuk mendapatkan individu`Upload`objek untuk setiap file yang ditransfer. Untuk informasi selengkapnya, lihatDapatkan Kemajuan Subtransfer.

LihatLengkapi Contohdi 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 menyediakanAmazon S3nama ember, key prefix S3, aBerkasobjek yang mewakili direktori lokal untuk menyalin, dan`boolean`nilai yang menunjukkan apakah Anda ingin menyalin subdirektori secara rekursif (benarataupalsu).

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();
```

LihatTunggu Transfer untuk Menyelesaikanuntuk 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. LihatDapatkan Status Transfer dan Kemajuanuntuk informasi lebih lanjut.

ParameterMultipleFileUploadobjek yang dikembalikan oleh`uploadFileList`dapat digunakan untuk query negara transfer atau kemajuan. LihatPoll Kemajuan Transfer Saat InidanDapatkan Transfer Progress dengan ProgressListeneruntuk informasi lebih lanjut.

Anda juga dapat menggunakan`MultipleFileUpload`'s`getSubTransfers`metode untuk mendapatkan individu`Upload`objek untuk setiap file yang ditransfer. Untuk informasi selengkapnya, lihatDapatkan Kemajuan Subtransfer.

LihatLengkapi Contohdi GitHub.

Mengunduh File atau Direktori

Menggunakan TransferManager kelas untuk men-download salah satu file (Amazon S3objek) atau direktori (sebuahAmazon S3nama bucket diikuti oleh awalan objek) dariAmazon S3.

Topik

- Mengunduh File Tunggal

- Unduh Direktori

Mengunduh File Tunggal

Gunakan TransferManager`download`metode, menyediakanAmazon S3nama bucket yang berisi objek yang ingin Anda download, kunci (objek) nama, danBerkasobjek 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.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

LihatTunggu Transfer untuk Menyelesaikanuntuk 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. LihatDapatkan Status Transfer dan Kemajuanuntuk informasi lebih lanjut.

LihatLengkapi Contohdi GitHub.

Unduh Direktori

Untuk mengunduh satu set file yang berbagi key prefix umum (analog dengan direktori pada sistem file) dariAmazon S3, gunakan TransferManager`downloadDirectory`metode. Metode ini mengambilAmazon S3nama bucket yang berisi objek yang ingin Anda download, awalan objek bersama oleh semua objek, danBerkasobjek 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.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

LihatTunggu Transfer untuk Menyelesaikanuntuk 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. LihatDapatkan Status Transfer dan Kemajuanuntuk informasi lebih lanjut.

LihatLengkapi Contohdi 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.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

LihatLengkapi Contohdi GitHub.

Tunggu Transfer untuk Menyelesaikan

Jika aplikasi Anda (atau thread) dapat memblokir sampai transfer selesai, Anda dapat menggunakantransferantarmuka`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 acarasebelumnyapanggilan`waitForCompletion`, menerapkan mekanisme pemungutan suara pada thread terpisah, atau menerima pembaruan kemajuan secara asinkron menggunakanProgressListener.

LihatLengkapi Contohdi 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 menerapkantransferantarmuka.`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

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);
```

LihatLengkapi Contohdi GitHub.

Dapatkan Transfer Progress dengan ProgressListener

Anda dapat melampirkanProgressListenerke transfer apa pun dengan menggunakantransferAntarmukaaddProgressListenermetode.

SEBUAHProgressListenerhanya membutuhkan satu metode,progressChanged, yang mengambilProgressEventobjek. Anda dapat menggunakan objek untuk mendapatkan total byte operasi dengan memanggil nyagetBytesmetode, dan jumlah byte ditransfer sejauh ini dengan memanggilgetBytesTransferred.

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();
```

LihatLengkapi Contohdi GitHub.

Dapatkan Kemajuan Subtransfer

ParameterMultipleFileUploadkelas dapat mengembalikan informasi tentang subtransfernya dengan memanggil nya`getSubTransfers`metode. Ia mengembalikan sebuah tidak dapat dimodifikasiKoleksidariUnggahobjek 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);
```

LihatLengkapi Contohdi GitHub.

Info Selengkapnya

- Kunci objekdiAmazon Simple Storage ServicePanduan Pengguna

MengonfigurasiAmazon S3Bucket sebagai Situs Web

Anda dapat mengonfigurasiAmazon S3bucket untuk berperilaku sebagai situs web. Untuk melakukan ini, Anda perlu mengatur konfigurasi situs webnya.

> **Note**
>
> Contoh-contoh kode ini berasumsi bahwa Anda memahami materi diMenggunakanAWS SDK for Javadan telah dikonfigurasi defaultAWSkredensi menggunakan informasi diMengaturAWSKredensial dan Wilayah untuk Pembangunan.

Mengatur Konfigurasi Situs Web Bucket

Untuk mengaturAmazon S3konfigurasi situs bucket, hubungi AmazonS3`setWebsiteConfiguration`metode dengan nama bucket untuk mengatur konfigurasi untuk, danBucketWebsiteConfigurationobjek yang berisi konfigurasi website bucket.

Mengatur dokumen indeksdibutuhkan; 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.getErrorMessage());
    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 mengaturkebijakan bucketyang memungkinkan akses baca publik ke file dalam bucket. Untuk informasi selengkapnya, lihatMengelola Akses keAmazon S3ember menggunakan kebijakan bucket.

LihatLengkapi Contohdi GitHub.

Dapatkan Konfigurasi Situs Web Bucket

Untuk mendapatkanAmazon S3konfigurasi situs bucket, hubungi AmazonS3`getWebsiteConfiguration`metode dengan nama bucket untuk mengambil konfigurasi untuk.

Konfigurasi akan dikembalikan sebagaiBucketWebsiteConfigurationobjek. Jika tidak ada konfigurasi situs web untuk bucket, makan`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.getErrorMessage());
    System.out.println("Failed to get website configuration!");
    System.exit(1);
}
```

LihatLengkapi Contohdi GitHub.

Menghapus Konfigurasi Situs Web Bucket

Untuk menghapusAmazon S3konfigurasi 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.getErrorMessage());
    System.out.println("Failed to delete website configuration!");
    System.exit(1);
}
```

LihatLengkapi Contohdi GitHub.

Informasi Selengkapnya

- Situs web Bucket PUTdiAmazon S3Referensi API
- DAPATKAN situs web BucketdiAmazon S3Referensi API

- BERHAPUS situs web BucketdiAmazon S3Referensi API

GunakanAmazon S3Enkripsi sisi klien

Mengenkripsi data menggunakanAmazon S3enkripsi klien adalah salah satu cara Anda dapat memberikan lapisan perlindungan tambahan untuk informasi sensitif yang Anda simpan diAmazon S3. Contoh dalam bagian ini menunjukkan cara membuat dan mengonfigurasiAmazon S3enkripsi klien untuk aplikasi Anda.

Jika Anda baru mengenal kriptografi, lihatDasar KriptografidiAWSPanduan Pengembang KMS untuk gambaran dasar istilah kriptografi dan algoritma. Untuk informasi tentang dukungan kriptografi di semuaAWSSDK, lihatAWSSupport SDK untukAmazon S3Enkripsi sisi klendiAmazon Web ServicesReferensi Umum.

> **Note**
>
> Contoh-contoh kode ini berasumsi bahwa Anda memahami materi diMenggunakanAWS SDK for Javadan telah dikonfigurasi defaultAWSkredensi menggunakan informasi diMengaturAWSKredensial dan Wilayah untuk Pembangunan.

Jika Anda menggunakan versi 1.11.836 atau versi sebelumnyaAWS SDK for Java, lihatAmazon S3Migrasi Klienuntuk informasi tentang migrasi aplikasi Anda ke versi yang lebih baru. Jika Anda tidak dapat bermigrasi, lihatcontoh lengkap inidi GitHub.

Jika tidak, jika Anda menggunakan versi 1.11.837 atau yang lebih baru dariAWS SDK for Java, jelajahi contoh topik yang tercantum di bawah ini untuk digunakanAmazon S3enkripsi di sisi klien.

Topik

- Amazon S3enkripsi sisi klien
- Amazon S3enkripsi sisi klien denganAWSKunci terkelola KMS

Amazon S3enkripsi sisi klien

Contoh berikut menggunakanAmazons3EncryptionClientV2Builderkelas untuk membuatAmazon S3enkripsi sisi klien diaktifkan. Setelah diaktifkan, objek apa pun yang Anda unggahAmazon S3menggunakan klien ini akan dienkripsi. Setiap benda yang Anda dapatkan dariAmazon S3menggunakan klien ini akan secara otomatis didekripsi.

> **Note**
>
> Contoh berikut menunjukkan menggunakanAmazon S3enkripsi sisi klien dengan kunci master klien. Untuk mempelajari cara menggunakan enkripsi denganAWSKunci terkelola KMS, lihatAmazon S3enkripsi sisi klien denganAWSKunci terkelola KMS.

Anda dapat memilih dari dua mode enkripsi saat mengaktifkan sisi klienAmazon S3enkripsi: ketat dikonfirmasi atau diautentikasi. Bagian berikut menunjukkan cara mengaktifkan setiap jenis. Untuk mempelajari algoritma yang digunakan setiap mode, lihatCryptoModedefinisi.

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`StrictAuthenticatedEncryption`value dalam`withCryptoConfiguration`metode.

> **Note**
>
> Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan yang terbaruGoyang Puri jarfile 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`AuthenticatedEncryption`mode, 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, lihatAmazon S3Enkripsi terautentikasiposting blog.

> **Note**
>
> Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan yang terbaruGoyang Puri jarfile di classpath aplikasi Anda.

Untuk mengaktifkan mode ini, tentukan`AuthenticatedEncryption`value 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 menggunakanAmazons3EncryptionClientV2Builderkelas untuk membuatAmazon S3enkripsi di sisi klien diaktifkan. Setelah dikonfigurasi, objek apa pun yang Anda unggahAmazon S3menggunakan klien ini akan dienkripsi. Setiap benda yang Anda dapatkan dariAmazon S3menggunakan klien ini secara otomatis didekripsi.

> **Note**
>
> Contoh berikut menunjukkan cara menggunakanAmazon S3enkripsi sisi klien denganAWSKMS kunci yang dikelola. Untuk mempelajari cara menggunakan enkripsi dengan kunci Anda sendiri, lihatAmazon S3enkripsi sisi klien.

Anda dapat memilih dari dua mode enkripsi saat mengaktifkan sisi klienAmazon S3enkripsi: ketat dikonfirmasi atau diautentikasi. Bagian berikut menunjukkan cara mengaktifkan setiap jenis. Untuk mempelajari algoritma yang digunakan setiap mode, lihatCryptoModedefinisi.

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 terbaruGoyang 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 metodeAmazon S3enkripsi 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`AuthenticatedEncryption`mode, 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, lihatAmazon S3Enkripsi terautentikasi sisi klienposting blog.

> **Note**
>
> Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan yang terbaruGoyang Puri jarfile di classpath aplikasi Anda.

Untuk mengaktifkan mode ini, tentukan`AuthenticatedEncryption`value dalam`withCryptoConfiguration`metode.

Kode

```
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
         .withRegion(Regions.US_WEST_2)
         .withCryptoConfiguration(new
 CryptoConfigurationV2().withCryptoMode((CryptoMode.AuthenticatedEncryption)))
         .withEncryptionMaterialsProvider(new KMSEncryptionMaterialsProvider(keyId))
         .build();
```

MengonfigurasiAWS KMSklien

ParameterAmazon S3enkripsi klien menciptakanAWS KMSklien secara default, kecuali salah satu secara eksplisit ditentukan.

Untuk mengatur wilayah untuk ini dibuat secara otomatisAWS KMSklien, 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 sendiriAWS KMSklien 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 SQSContoh MenggunakanAWS SDK for Java

Bagian ini menyediakan contoh pemrogramanAmazon SQSmenggunakanAWS SDK for Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk menunjukkan setiap teknik. Parameterkode 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 denganAmazon SQSAntrean pesan
- Mengirim, Menerima, dan MenghapusAmazon SQSPesan
- Mengaktifkan Jajak pendapat panjangAmazon SQSAntrean
- Mengatur Timeout Visibilitas diAmazon SQS
- Menggunakan Antrian Surat Mati diAmazon SQS

Bekerja denganAmazon SQSAntrean pesan

SEBUAHAntrean pesanadalah wadah logis yang digunakan untuk mengirim pesan andalAmazon SQS. Ada dua tipe antrian:standardanpertama-dalam, pertama-keluar(FIFO). Untuk mempelajari lebih lanjut tentang antrian dan perbedaan antara jenis ini, lihatAmazon SQSPanduan Pengembang.

Topik ini menjelaskan cara membuat, membuat daftar, menghapus, dan mendapatkan URLAmazon SQSantrian dengan menggunakanAWS SDK for Java.

Membuat Antrean

Gunakan klien AmazonSqs`createQueue`metode, menyediakanCreateQueueRequestobjek 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(QUEUE_NAME)
        .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());
```

LihatLengkapi Contohdi GitHub.

Antrean

Untuk membuat daftarAmazon SQSantrian 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 kembaliAntrean. 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);
}
```

LihatLengkapi Contohdi GitHub.

Mendapatkan URL untuk Antrian

Hubungi klien Amazon Sqs`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(QUEUE_NAME).getQueueUrl();
```

LihatLengkapi Contohdi GitHub.

Menghapus Antrean

Berikan antreanURLke klien Amazon Sqs`deleteQueue`metode.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
sqs.deleteQueue(queue_url);
```

LihatLengkapi Contohdi GitHub.

Info Selengkapnya

- BagaimanaAmazon SQSAntreandiAmazon SQSPanduan Pengembang
- CreateQueuediAmazon SQSReferensi API
- GetQueueUrldiAmazon SQSReferensi API
- ListQueuesdiAmazon SQSReferensi API

- DeleteQueuesdiAmazon SQSReferensi API

Mengirim, Menerima, dan MenghapusAmazon SQSPesan

Topik ini menjelaskan cara mengirim, menerima, dan menghapusAmazon SQSpesan. Pesan selalu dikirimkan menggunakanAntrean SQS.

Mengirim pesan

Menambahkan satu pesan keAmazon SQSantrian dengan memanggil klien AmazonSqs`sendMessage`metode. BerikanSendMessageRequestobjek yang berisi antrianURL, 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);
```

LihatLengkapi Contohdi 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 mengambilSendMessageBatchRequestberisi URL antrian dan daftar pesan (masing-masing aSendMessageBatchRequestEntryUntuk 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);
```

LihatLengkapi Contohdi GitHub.

Menerima Pesan

Mengambil setiap pesan yang saat ini dalam antrian dengan memanggil klien AmazonSqs`receiveMessage`metode, melewatinya URL antrian ini. Pesan dikembalikan sebagai daftarPesanbenda.

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());
```

```
}
```

LihatLengkapi Contohdi GitHub.

Info Selengkapnya

- BagaimanaAmazon SQSAntrean KerjadiAmazon SQSPanduan Pengembang
- SendMessagediAmazon SQSReferensi API
- SendMessageBatchdiAmazon SQSReferensi API
- ReceiveMessagediAmazon SQSReferensi API
- DeleteMessagediAmazon SQSReferensi API

Mengaktifkan Jajak pendapat panjangAmazon SQSAntrean

Amazon SQSmenggunakanJajak pendapatsecara 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 penggunaanAmazon SQSdengan mengurangi jumlah tanggapan kosong ketika tidak ada pesan yang tersedia untuk kembali sebagai balasan ReceiveMessage permintaan dikirim keAmazon SQSantrian dan menghilangkan tanggapan kosong palsu.

> **Note**
>
> Anda dapat mengatur frekuensi polling panjang dari1-20 detik.

Mengaktifkan Polling Panjang saat Membuat Antrian

Untuk mengaktifkan polling panjang saat membuatAmazon SQSantrian, mengaturReceiveMessageWaitTimeSecondsAtribut padaCreateQueueRequestobjek sebelum memanggil kelas AmazonSqscreateQueuemetode.

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;
    }
}
```

LihatLengkapi Contohdi 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`padaSetQueueAttributesRequestsebelum 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);
```

LihatLengkapi Contohdi GitHub.

Mengaktifkan Pemungutan Suara Panjang pada Tanda Terima Pesan

Anda dapat mengaktifkan polling panjang saat menerima pesan dengan mengatur waktu tunggu dalam hitungan detik padaReceiveMessageRequestyang Anda suplai ke kelas AmazonSqs`receiveMessage`metode.

> **Note**
>
> Anda harus memastikan bahwaAWSBatas 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);
```

LihatLengkapi Contohdi GitHub.

Info Selengkapnya

- Amazon SQSJajak pendapatdiAmazon SQSPanduan Pengembang
- CreateQueuediAmazon SQSReferensi API
- ReceiveMessagediAmazon SQSReferensi API
- setQueueAttributesdiAmazon SQSReferensi API

Mengatur Timeout Visibilitas diAmazon SQS

Ketika pesan diterima diAmazon SQS, tetap pada antrian sampai dihapus untuk memastikan penerimaan. Sebuah pesan yang diterima, tetapi tidak dihapus, akan tersedia dalam permintaan

berikutnya setelah diberikanbatas waktu visibilitasuntuk membantu mencegah pesan diterima lebih dari sekali sebelum dapat diproses dan dihapus.

> **Note**
>
> Saat menggunakanAntrean 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 dalamChangeMessageVisibilityRequestyang 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);
```

Lihatcontoh lengkapdi GitHub.

Mengatur Timeout Visibilitas Pesan untuk Beberapa Pesan sekaligus

Untuk mengatur batas waktu tunggu visibilitas pesan untuk beberapa pesan sekaligus, buat daftarChangeMessageVisibilityBatchRequestEntryobjek, masing-masing berisi string ID unik dan pegangan tanda terima. Kemudian, lewati daftar keAmazon SQSclass 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);
```

Lihatcontoh lengkapdi GitHub.

Info Selengkapnya

- Timeout Visibilitasdi dalamAmazon SQSPanduan Pengembang
- setQueueAttributesdi dalamAmazon SQSReferensi API
- GetQueueAttributesdi dalamAmazon SQSReferensi API
- ReceiveMessagedi dalamAmazon SQSReferensi API

- ChangeMessageVisibilitydi dalamAmazon SQSReferensi API
- ChangeMessageVisibilityBatchdi dalamAmazon SQSReferensi API

Menggunakan Antrian Surat Mati diAmazon SQS

Amazon SQSmenyediakan dukungan untukAntrean surat mati. Antrean surat mati adalah antrean 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 samaAkun AWSdan wilayah sebagai antrian sumber.

Di sini kita membuat dua identikAmazon SQSantrian, 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;
    }
}
```

LihatLengkapi Contohdi GitHub.

Menunjuk Antrian Surat Mati untuk Antrian Sumber

Untuk menunjuk antrian surat mati, Anda harus terlebih dahulu membuatkebijakan 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 AmazonSqs`setQueueAttributes`Metode 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);
```

LihatLengkapi Contohdi GitHub.

Info Selengkapnya

- MenggunakanAmazon SQSAntrian Surat MatidiAmazon SQSPanduan Pengembang
- setQueueAttributesdiAmazon SQSReferensi API

Amazon SWFContoh MenggunakanAWS SDK for Java

Amazon SWFadalah 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 bahkanLambdatugas.

Ada dua cara untuk bekerja denganAmazon SWFmenggunakanAWS SDK for Java, dengan menggunakan SWFklienobjek, atau dengan menggunakanAWS Flow Frameworkuntuk Java. ParameterAWS Flow Frameworkuntuk 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 menggunakanAWS Flow Frameworkuntuk Java. Untuk informasi selengkapnya, lihatAWS Flow Frameworkuntuk Panduan Developer Java.

Bagian ini menyediakan contoh pemrogramanAmazon SWFdengan menggunakanAWS SDK for Javaklien 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 denganAmazon SWFmenggunakanAWS SDK for Java. Hal ini dimaksudkan terutama untuk referensi. Untuk tutorial pengantar yang lebih lengkap, lihatMembangun SederhanaAmazon SWFAplikasi.

Dependensi

DasarAmazon SWFaplikasi akan membutuhkan dependensi berikut, yang disertakan denganAWS SDK for Java:

- aws-java-sdk-1.12.*.jar
- umum-loging-1.2.*.jar
- httpclient-4.3.*.jar
- httpcore-4.3.*.jar
- jackson-anotasi-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 Frameworkuntuk aplikasi Java memerlukan pengaturan tambahan,dandependensi tambahan. LihatAWS Flow Frameworkuntuk Panduan Developer JavaUntuk 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 di`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 menggunakanAWS Flow Frameworkuntuk 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**
>
> ParameterAWS Flow Frameworkuntuk Java memiliki persyaratan tambahan di luar basisAWS SDK for Java. Untuk informasi lebih lanjut, lihatAWS Flow Frameworkuntuk Panduan Developer Java.

Menggunakan kelas klien SWF

Antarmuka dasar Anda untukAmazon SWFadalah melalui salah satuAmazonSimpleWorkflowClientatauAmazonSimpleWorkflowAsyncClientkelas. Perbedaan utama antara ini adalah bahwa`\*AsyncClient`Kelas kembaliMasa Depanobjek untuk pemrograman bersamaan (asinkron).

```
AmazonSimpleWorkflowClient swf = AmazonSimpleWorkflowClientBuilder.defaultClient();
```

Membangun Amazon SWF Aplikasi Sederhana

Topik ini akan memperkenalkan Anda ke Amazon SWFaplikasi pemrograman denganAWS 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 tradisiHelloWorld, 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:

- Itu AWS SDK for Java.
- Apache Maven (3.3.1).
- JDK 1.7 atau yang lebih baru. Tutorial ini dikembangkan dan diuji menggunakan JDK 1.8.0.
- Editor teks Java yang bagus (pilihan Anda).

> **Note**
>
> Jika Anda menggunakan sistem build yang berbeda dari Maven, Anda masih dapat membuat proyek menggunakan langkah-langkah yang sesuai untuk lingkungan Anda dan menggunakan konsep yang disediakan di sini untuk mengikuti. Informasi lebih lanjut tentang mengkonfigurasi dan menggunakan AWS SDK for Java dengan berbagai sistem build disediakan di Memulai.
> Demikian juga, tetapi dengan lebih banyak usaha, langkah-langkah yang ditunjukkan di sini dapat diimplementasikan menggunakan salah satu AWS SDK dengan dukungan untuk. Amazon SWF

Semua dependensi eksternal yang diperlukan disertakan denganAWS SDK for Java, jadi tidak ada tambahan untuk diunduh.

AWSAkses

Agar berhasil mengerjakan tutorial ini, Anda harus memiliki akses ke portal AWS akses seperti yang dijelaskan di bagian penyiapan dasar panduan ini.

Petunjuk menjelaskan cara mengakses kredensyal sementara yang Anda salin dan tempel ke file bersama `credentials` lokal Anda. Kredensyal sementara yang Anda tempel harus dikaitkan dengan peran IAM AWS IAM Identity Center yang memiliki izin untuk mengakses Amazon SWF. Setelah menempelkan kredensi sementara, `credentials` file Anda akan terlihat serupa dengan yang berikut ini.

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONGSTRI
```

Kredensi sementara ini terkait dengan profil. `default`

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-simpleworkflowmodul 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 AmazonSimpleWorkflowClientkelas 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 AmazonSimpleWorkflowClientkelas. Ini adalah antarmuka dasar untuk Amazon SWF metode yang disediakan olehAWS SDK for Java.

```
private static final AmazonSimpleWorkflow swf =

 AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

   Cuplikan sebelumnya mengasumsikan bahwa kredensyal 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 RegisterDomainRequestuntuk informasi lebih lanjut.

   Jika domain dengan nama itu sudah ada, a akan DomainAlreadyExistsExceptiondinaikkan. 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. `0with*`

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 RegisterActivityTypeRequestuntuk informasi lebih lanjut.

> **Note**
>
> Semua nilai batas waktu ditentukan dalam hitungan detik. Lihat Amazon SWFTimeout 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 RegisterWorkflowTypeRequestuntuk informasi lebih lanjut.

+

Jika jenis alur kerja yang Anda coba daftarkan sudah ada, sebuah akan TypeAlreadyExistsExceptiondinaikkan. 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 file`ActivityWorker.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 denganAmazon 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 passsed-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 tidak`null`, 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 RespondActivityTaskCompletedRequestobjek yang berisi token tugas dan data hasil aktivitas.

Di sisi lain, jika tugas gagal, maka kita merespons dengan memanggil `respondActivityTaskFailed` metode dengan RespondActivityTaskFailedRequestobjek, 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 file`WorkflowWorker.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 AmazonSimpleWorkflowClientkelas 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` PollForDecisionTaskRequestMemberikan 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 HistoryEventobjek yang dikirim dengan tugas.

5. Tambahkan `executeDecisionTask` metode, mengambil token tugas (a`String`) 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 HistoryEventkelas untuk informasi tentang setiap jenis acara.

+ CATATAN: String dalam `switch` pernyataan diperkenalkan di Jawa 7. Jika Anda menggunakan versi Java sebelumnya, Anda dapat menggunakan EventTypekelas 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 ScheduleActivityTaskDecisionAttributesstruktur 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

CompleteWorkflowExecutionDecisionAttributesstruktur 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 file`WorkflowStarter.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() + "'.");
    }
}
```

`WorkflowStarter`Kelas terdiri dari metode tunggal,`main`, yang mengambil argumen opsional diteruskan pada baris perintah sebagai data input untuk alur kerja.

Metode klien SWF,`startWorkflowExecution`, mengambil StartWorkflowExecutionRequestobjek 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 olehAmazon 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 lanjutAmazon SWF, kunjungi Amazon SWFhalaman 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 SWFkegiatan, Anda dapat menggunakanLambdaberfungsi untuk mewakili unit kerja dalam alur kerja Anda, dan menjadwalkan mereka sama dengan kegiatan.

Topik ini berfokus pada bagaimana menerapkanAmazon SWF Lambdatugas menggunakanAWS SDK for Java. Untuk informasi lebih lanjut tentangLambdatugas secara umum, lihatAWS LambdaTugasdiAmazon SWFPanduan Developer.

Menyiapkan IAM role untuk menjalankan fungsi Lambda

SebelumAmazon SWFdapat menjalankanLambdafungsi, Anda perlu mengatur peran IAM untuk memberikanAmazon SWFizin untuk menjalankanLambdafungsi atas nama Anda. Untuk informasi lengkap tentang cara melakukan ini, lihatAWS LambdaTugas.

Anda memerlukan Amazon Resource Name (ARN) dari IAM role ini ketika mendaftarkan alur kerja yang akan digunakanLambdatugas.

BuatLambdafungsi

Anda dapat menulisLambdafungsi dalam sejumlah bahasa yang berbeda, termasuk Java. Untuk informasi lengkap tentang cara menulis, menyebarkan, dan menggunakanLambdafungsi, lihatAWS 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 diMembangun SederhanaAmazon SWFAplikasi.

- Versi ini ditulis dalam JavaScript, yang dapat dimasukkan langsung menggunakanAWS 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, lihatMembuat Paket Deployment (Java)diAWS LambdaPanduan Developer. Anda juga akan ingin melihat bagian berjudulModel Pemrograman untuk AuthoringLambdaFungsi di Java.

Lambdamengambilperistiwaataumemasukkanobjek sebagai parameter pertama, dankonteksobjek sebagai yang kedua, yang memberikan informasi tentang permintaan untuk menjalankanLambdafungsi. Fungsi khusus ini mengharapkan masukan untuk berada di JSON, denganwhobidang diatur ke nama yang digunakan untuk membuat ucapan.

Daftarkan alur kerja untuk digunakan dengan Lambda

Untuk alur kerja untuk menjadwalkanLambdafungsi, Anda harus memberikan nama peran IAM yang menyediakanAmazon SWFdengan izin untuk memohonLambdafungsi. Anda dapat mengatur ini selama pendaftaran alur kerja dengan menggunakan`withDefaultLambdaRole`atau`setDefaultLambdaRole`metodeRegisterWorkflowTypeReq

```
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 memberikanKeputusandengan `ScheduleLambdaFunction`DecisionTypedan dengandenganScheduleLambdaFunctionDecisionAttributes.

```
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`ScheduleLambdaFuntionDecisionAttributes`, Anda harus menyediakannama, yang merupakan ARN dariLambdafungsi untuk memanggil, danid, yang merupakan nama yangAmazon SWFakan digunakan untuk mengidentifikasiLambdafungsi dalam log sejarah.

Anda juga dapat memberikan opsionalmemasukkanuntukLambdafungsi dan mengatur nyamulai menutup batas waktunilai, yang merupakan jumlah detik yangLambdafungsi diperbolehkan untuk menjalankan sebelum menghasilkan`LambdaFunctionTimedOut`peristiwa.

> **Note**
>
> Kode ini menggunakanAWSLambdaClientuntuk mengambil ARN dariLambdafungsi, mengingat nama fungsi. Anda dapat menggunakan teknik ini untuk menghindari hard-coding ARN penuh (yang mencakup AndaAkun AWSID) 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, denganEventTypenilai-nilai sepertiLambdaFunctionScheduled,LambdaFunctionStarted, danLambdaFunctionCompleted. JikaLambdafungsi gagal, atau membutuhkan waktu lebih lama untuk menjalankan dari nilai batas waktu yang ditetapkan, Anda akan menerima baikLambdaFunctionFailedatauLambdaFunctionTimedOutjenis 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 dariLambdafungsi

Ketika Anda menerimaLambdaFunctionCompleted`EventType, you can retrieve your 0 function's return value by first calling `getLambdaFunctionCompletedEventAttributespadaHistoryEventuntuk mendapatkanLambdaFunctionCompletedEventAttributesobjek, dan kemudian memanggil nyagetResultmetode untuk mengambil output dariLambdaFungsi:

```
 LambdaFunctionCompleted:
running_functions--;
```

Sumber lengkap untuk contoh ini

Anda dapat menelusurisumber lengkap:github: `<awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf_lambda/>untuk contoh ini di Github diaws-java-pengembang-panduanrepositori.

Mematikan Kegiatan dan Alur Kerja Pekerja Anggun

ParameterMembangun SederhanaAmazon SWFAplikasitopik 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 olehAmazon SWFuntuk menjalankan kegiatan atau keputusan kembali. Setelah permintaan jajak pendapat dibuat,Amazon SWFmencatat poller dan akan mencoba untuk menetapkan tugas untuk itu.

Jika pekerja alur kerja dihentikan selama polling panjang,Amazon SWFmungkin 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 darihelloswf, 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()
```

```
                            .withTaskToken(taskToken)
                            .withReason(error.getClass().getSimpleName())
                            .withDetails(error.getMessage()));
                }
            }
        }
    }
}
```

Dalam versi ini, kode pemungutan suara yang ada di`main`fungsi dalam versi asli telah dipindahkan ke metode sendiri,`pollAndExecute`.

Parameter`main`fungsi sekarang menggunakanCountdownLatchdalam hubungannya dengankait penghentianmenyebabkan benang menunggu hingga 60 detik setelah penghentiannya diminta sebelum membiarkan benang ditutup.

Mendaftarkan domain

Setiap alur kerja dan aktivitas diAmazon SWFmembutuhkanranahuntuk menjalankan di.

1. Membuat baruRegisterDomainRequestobjek, menyediakan dengan setidaknya nama domain dan periode retensi eksekusi alur kerja (parameter ini keduanya diperlukan).
2. MemanggilAmazonSimpleWorkflowClient.registerDomainmetode denganRegisterDomainRequestobjek.
3. TangkapDomainAlreadyExistsExceptionjika domain yang Anda minta sudah ada (dalam hal ini, tidak ada tindakan yang biasanya diperlukan).

Kode berikut menunjukkan prosedur ini:

```
public void register_swf_domain(AmazonSimpleWorkflowClient swf, String name)
{
    RegisterDomainRequest request = new RegisterDomainRequest().withName(name);
    request.setWorkflowExecutionRetentionPeriodInDays("10");
    try
    {
        swf.registerDomain(request);
    }
    catch (DomainAlreadyExistsException e)
    {
        System.out.println("Domain already exists!");
```

```
    }
}
```

Daftar domain

Anda dapat daftarAmazon SWFdomain yang terkait dengan akun Anda danAWSwilayah berdasarkan jenis registrasi.

1. BuatListDomainsRequestobjek, dan tentukan status pendaftaran domain yang Anda minati—ini diperlukan.
2. PANGLANAmazonSimpleWorkflowClient.ListDomainsdenganListDomainRequestobjek. Hasil disediakan dalamDomainInfosobjek.
3. PANGLANGetDomainInfospada objek kembali untuk mendapatkan daftarDomainInfoobjek.
4. PANGLANgetNamepada masing-masingDomainInfokeberatan 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

ParameterAWS SDK for Javahadir dikemas dengan sampel kode yang menunjukkan banyak fitur SDK dalam program buildable dan runnable. Anda dapat mempelajari atau memodifikasi ini untuk mengimplementasikan Anda sendiriAWSsolusi menggunakanAWS SDK for Java.

Cara Mendapatkan Sampel

ParameterAWS SDK for Javasampel kode disediakan dalammencicipidirektori SDK. Jika Anda mengunduh dan menginstal SDK menggunakan informasi diMenyiapkanAWS SDK for Java, Anda sudah memiliki sampel pada sistem Anda.

Anda juga dapat melihat sampel terbaru diAWS SDK for JavaRepositori GitHub, disrc/sampeldirektori.

Membangun dan Menjalankan Sampel Menggunakan Command Line

Sampel termasukSemutmembangun 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, klikMentahtombol 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 satuAWS SDK for Javasampel, Anda perlu mengaturAWSkredensi di lingkungan atau denganAWS CLI, seperti yang ditentukan dalamMengaturAWSKredensial dan Wilayah untuk Pembangunan. Sampel menggunakan rantai penyedia kredenensi default bila memungkinkan. Jadi dengan menetapkan kredensi Anda dengan cara ini, Anda dapat menghindari praktik berisiko memasukkan AndaAWSkredensi 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 rootAWSSDK 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 menggunakanAWS Toolkit for Eclipse, Anda juga dapat memulai proyek baru di Eclipse berdasarkanAWS SDK for Javaatau tambahkan SDK ke proyek Java yang ada.

Prasyarat

Setelah menginstalAWS Toolkit for Eclipse, kami sarankan untuk mengonfigurasi Toolkit dengan kredensi keamanan Anda. Anda dapat melakukannya kapan saja dengan memilihPreferensidariJendelamenu di Eclipse, dan kemudian memilih AWSToolkitbagian.

Menjalankan Sampel

1. Buka Eclipse.
2. MembuatAWSProyek Java. Dalam Eclipse, diBerkasmenu, pilihBaru, dan kemudian klikProyek. ParameterProyek BaruPenyihir terbuka.
3. Perluas AWS kategori, lalu pilih AWSProyek Java.
4. Pilih Selanjutnya. Halaman pengaturan proyek ditampilkan.
5. Masukkan nama diNama Proyekkotak. ParameterAWS SDK for JavaKelompok 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. MasukkanAWSkredenensi. Jika Anda sudah mengkonfigurasiAWS Toolkit for Eclipsedengan kredensi Anda, ini secara otomatis diisi.
8. Pilih Selesai. Proyek ini dibuat dan ditambahkan keExplorer Proyek.
9. Pilih sampel`.java`file yang ingin Anda jalankan. Misalnya, untukAmazon S3sampel, pilih`S3Sample.java`.
10. PilihJalankandariJalankanmenu.
11. Klik kanan proyek diExplorer Proyek, arahkan keJalur Bangun, dan kemudian pilihTambahkan Pustaka.
12. Pilih AWSSDK Java, pilihSelanjutnya, lalu ikuti petunjuk di layar yang tersisa.

Keamanan untuk AWS SDK for Java

Keamanan cloud di Amazon Web Services (AWS) merupakan prioritas tertinggi. Sebagai pelanggan AWS , Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan. Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model Tanggung Jawab Bersama menggambarkan ini sebagai Keamanan dari Cloud dan Keamanan dalam Cloud.

Security of the Cloud - AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan semua layanan yang ditawarkan di AWS Cloud dan memberi Anda layanan yang dapat Anda gunakan dengan aman. Tanggung jawab keamanan kami adalah prioritas tertinggi di AWS, dan efektivitas keamanan kami secara teratur diuji dan diverifikasi oleh auditor pihak ketiga sebagai bagian dari Program AWS Kepatuhan.

Keamanan di Cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan, dan faktor-faktor lain termasuk sensitivitas data Anda, persyaratan organisasi Anda, serta undang-undang dan peraturan yang berlaku.

AWS Produk atau layanan ini mengikuti model tanggung jawab bersama melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat halaman dokumentasi keamanan AWS layanan dan AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan.

Topik

Perlindungan data 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 pengelolaan 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 kredensyal 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 dalam 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 kredensyal 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().getProto
```

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/

Menegakkan 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.

Manajemen Identitas dan Akses

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. AWS IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

Audiens

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 kredensyal dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak AWS fitur untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur AWS, lihat Memecahkan masalah AWS identitas dan akses atau panduan pengguna yang Layanan AWS Anda gunakan.

Administrator layanan — Jika Anda bertanggung jawab atas AWS sumber daya di perusahaan Anda, Anda mungkin memiliki akses penuh ke AWS. Tugas Anda adalah menentukan AWS fitur dan sumber daya mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM AWS, lihat panduan pengguna yang Layanan AWS Anda gunakan.

Administrator IAM – Jika Anda adalah administrator IAM, Anda mungkin ingin belajar dengan lebih detail tentang cara Anda menulis kebijakan untuk mengelola akses ke AWS. Untuk melihat contoh kebijakan AWS berbasis identitas yang dapat Anda gunakan di IAM, lihat panduan pengguna yang Anda gunakan. Layanan AWS

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal 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 kredensyal yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensyal Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas gabungan, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensyal Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar tugas lengkap yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensyal 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 kredensyal yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensyal sementara.

Untuk pengelolaan akses terpusat, sebaiknya Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apa yang dimaksud Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya andalkan kredensial temporer, dan bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, sebaiknya rotasikan kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran tersebut dimaksudkan untuk dapat diambil oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang metode untuk menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi diautentikasi, identitas tersebut dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi sekumpulan izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengaitkan izin yang ditetapkan ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Contoh, ketika Anda melakukan panggilan dalam layanan, umumnya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memulai tindakan lain

di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.

- Peran IAM – Peran layanan adalah peran IAM yang diambil layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat mengambil peran untuk melakukan sebuah tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

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

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Ikhtisar kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat dilampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat Gambaran umum daftar kontrol akses (ACL) di Panduan Developer Layanan Penyimpanan Ringkas Amazon.

Tipe kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke sebagian atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit

di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Layanan AWS bekerja dengan IAM

Untuk mendapatkan tampilan tingkat tinggi tentang cara Layanan AWS bekerja dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Untuk mempelajari cara menggunakan yang spesifik Layanan AWS dengan IAM, lihat bagian keamanan dari Panduan Pengguna layanan yang relevan.

Memecahkan masalah AWS identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan AWS dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di AWS
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS sumber daya saya

Saya tidak berwenang untuk melakukan tindakan di AWS

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `awes:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes: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 `awes:GetWidget`.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran AWS.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di AWS. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS sumber daya saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mempelajari apakah AWS mendukung fitur ini, lihatBagaimana Layanan AWS bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Memberikan akses kepada pengguna eksternal yang sah (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Validasi Kepatuhan untuk AWS Produk atau Layanan ini

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

> **Note**
>
> Tidak semua memenuhi Layanan AWS syarat HIPAA. Untuk informasi selengkapnya, lihat Referensi Layanan yang Memenuhi Syarat HIPAA.

- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk mengevaluasi sumber daya AWS Anda dan memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.
- AWS Audit ManagerIni Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

AWS Produk atau layanan ini mengikuti model tanggung jawab bersama melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat halaman dokumentasi keamanan AWS layanan dan AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan.

Ketahanan untuk AWS Produk atau Layanan ini

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones.

Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan.

Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

AWS Produk atau layanan ini mengikuti model tanggung jawab bersama melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat halaman dokumentasi keamanan AWS layanan dan AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan.

Keamanan Infrastruktur untuk AWS Produk atau Layanan ini

AWS Produk atau layanan ini menggunakan layanan terkelola, dan karenanya dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS Produk atau Layanan ini melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan pengguna utama IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

AWS Produk atau layanan ini mengikuti model tanggung jawab bersama melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat halaman dokumentasi keamanan AWS layanan dan AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan.

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 Correttodan 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, pertama-tama perbarui kode Anda untuk menggunakan pembuat klien layanan baru. Kemudian berikan materi enkripsi menggunakan metode pada builder yang telah diganti namanya, dan konfigurasikan 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.

Konfigurasikan 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 `SecureRandom`
- `rangeGetMode` = `CryptoRangeGetMode.DISABLED`
- `unsafeUndecryptableObjectPassthrough` = `false`

Perhatikan bahwa EncryptionOnlytidak 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 kripto 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.

Konfigurasikan 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);
```

Konfigurasikan 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 CryptoRangeGetModedi 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-----

xsFNBFd1gAUBEACqbmmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYpOviSWsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKROmRODzUuaokLPo24pfm9bnr1RnRtwt5ktPAA5bM9ZZaGKriej
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
```

```
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+
hV6XulGAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGws
MDyHNqyJeYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KD
OSn5CbmXpAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVg
roUVtprsmHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs
/Hd981FdVghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0HOVWHwQsaW
jMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxpyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJBlkke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHODysrGVCLcmuinUBaNlHmLDcGY
XZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKMO+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv0OPCvO
WTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+OTa63/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: 6 Desember 2023

Januari 12, 2024

Tambahkan spanduk yang mengumumkan akhir dukungan untuk AWS SDK for Java v1.x.

Desember 6, 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 S3klien.

14 Apr 2017

- Membuat sejumlah pembaruan pada AWS SDK for Java bagian Amazon S3Contoh 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 EC2Contoh 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.

Mar 21, 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 VisibilitasAmazon 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 Javabagian 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

Des 16, 2016

- Menambahkan contoh topik baru untukDynamoDB: 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 SWFContoh Menggunakan AWS SDK for Java bagian telah diperbarui dengan konten baru:
  - Amazon SWFDasar-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 baruAmazon SWF.
  - Mematikan Aktivitas dan Alur Kerja Pekerja dengan Anggun - Menjelaskan bagaimana Anda dapat menutup kelas Amazon SWF pekerja dengan anggun menggunakan kelas konkurensi Java.

Feb 23, 2016

- Sumber untuk Panduan AWS SDK for Java Pengembang telah dipindahkan ke aws-java-developer-guide.

28 Des 2015

- Mengatur JVM TTL untuk Pencarian Nama DNS telah dipindahkan dari Advanced ke Using theAWS 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 Kredensil 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 kredensil 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 kredensil 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.