Panduan Pengembang untuk versi 1.x

AWS SDK untuk Java 1.x

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

Table of Contents

AWS SDK untuk Java 1.x mencapai end-of-support pada 31 Desember 2025. Kami menyarankan Anda bermigrasi ke AWS SDK for Java 2.xuntuk terus menerima fitur baru, peningkatan ketersediaan, dan pembaruan keamanan.

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

Panduan Pengembang - AWS SDK untuk Java 1.x

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

Versi 2 dari SDK dirilis

Lihatlah AWS SDK untuk Java 2.x baru di https://github.com/aws/aws-sdk-java-v2/. Ini mencakup banyak fitur yang ditunggu-tunggu, seperti cara untuk memasang implementasi HTTP. Untuk memulai, lihat Panduan Pengembang AWS SDK untuk Java 2.x.

Dokumentasi dan Sumber Daya Tambahan

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

- AWS SDK untuk Java Referensi API
- Blog pengembang Java
- Forum pengembang Java
- GitHub:
  - Sumber dokumentasi
  - Masalah dokumentasi
  - Sumber SDK
  - Masalah SDK
  - Sampel SDK
  - Saluran Gitter
- Sebuah Katalog Kode Sampel AWS
- @awsforjava (Twitter)
- catatan rilis

Dukungan Eclipse IDE

Jika Anda mengembangkan kode menggunakan Eclipse IDE, Anda dapat menggunakan AWS Toolkit for Eclipseuntuk menambahkan ke proyek Eclipse yang ada atau untuk membuat proyek baru. AWS SDK untuk Java AWS SDK untuk Java Toolkit ini juga mendukung pembuatan dan pengunggahan Lambda fungsi, meluncurkan dan memantau Amazon EC2 instance, mengelola IAM pengguna dan grup keamanan, editor AWS CloudFormation template, dan banyak lagi.

Lihat Panduan AWS Toolkit for Eclipse Pengguna untuk dokumentasi lengkap.

Mengembangkan Aplikasi untuk Android

Jika Anda seorang pengembang Android, Amazon Web Services menerbitkan SDK yang dibuat khusus untuk pengembangan Android: Amplify Android (SDK Seluler AWS untuk Android).

Melihat Riwayat Revisi SDK

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

Membangun Dokumentasi Referensi Java untuk versi SDK Sebelumnya

Referensi AWS SDK untuk Java API mewakili versi terbaru SDK versi 1.x. Jika Anda menggunakan versi versi 1.x versi sebelumnya, Anda mungkin ingin mengakses dokumentasi referensi SDK yang cocok dengan versi yang Anda gunakan.

Cara termudah untuk membangun dokumentasi adalah menggunakan alat build Maven Apache. Unduh dan instal Maven terlebih dahulu jika Anda belum memilikinya di sistem Anda, lalu gunakan instruksi berikut untuk membuat dokumentasi referensi.

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

```
mvn javadoc:javadoc
```

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

Memulai

Bagian ini memberikan informasi tentang cara menginstal, mengatur, dan menggunakan AWS SDK untuk Java.

Topik

Pengaturan dasar untuk bekerja dengan Layanan AWS

Ikhtisar

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

- Anda harus dapat masuk ke portal AWS akses yang tersedia di AWS IAM Identity Center.
- Izin peran IAM yang dikonfigurasi untuk SDK harus mengizinkan akses ke Layanan AWS yang dibutuhkan aplikasi Anda. Izin yang terkait dengan kebijakan PowerUserAccess AWS terkelola cukup untuk sebagian besar kebutuhan pengembangan.
- Lingkungan pengembangan dengan elemen-elemen berikut:
  - File konfigurasi bersama yang diatur dengan cara berikut:
    - `config`File tersebut berisi profil default yang menentukan file Wilayah AWS.
    - `credentials`File berisi kredensi 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

Saat Anda menggunakan IDE, Anda juga dapat mengintegrasikan AWS Toolkit s agar lebih mudah digunakan Layanan AWS. AWS Toolkit for EclipseIni adalah dua toolkit yang dapat Anda gunakan untuk pengembangan Java. AWS Toolkit for IntelliJ

> **Important**
>
> Petunjuk di bagian penyiapan ini mengasumsikan bahwa Anda atau organisasi menggunakan IAM Identity Center. Jika organisasi Anda menggunakan penyedia identitas eksternal yang bekerja secara independen dari IAM Identity Center, cari tahu bagaimana Anda bisa mendapatkan kredensi sementara untuk SDK for Java untuk digunakan. Ikuti petunjuk ini untuk menambahkan kredensi sementara ke file. `~/.aws/credentials`
> Jika penyedia identitas Anda menambahkan kredensi sementara secara otomatis ke `~/.aws/credentials` file, pastikan bahwa nama profil tersebut `[default]` sehingga Anda tidak perlu memberikan nama profil ke SDK atau. AWS CLI

Kemampuan masuk ke portal AWS akses

Portal AWS akses adalah lokasi web tempat Anda masuk secara manual ke Pusat Identitas IAM. Format URL adalah `d-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 autentikasi Pusat Identitas IAM di Panduan Referensi Alat AWS SDKs dan Alat. Jangan ikuti Langkah 2 karena AWS SDK untuk Java 1.x tidak mendukung penyegaran token otomatis dan pengambilan otomatis kredenal sementara untuk SDK yang dijelaskan Langkah 2.

Siapkan file konfigurasi bersama

File konfigurasi bersama berada di workstation pengembangan Anda dan berisi pengaturan dasar yang digunakan oleh semua AWS SDKs 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.

Siapkan **`config`** file bersama

Contoh berikut menunjukkan konten `config` file bersama.

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

Untuk tujuan pengembangan, gunakan yang Wilayah AWS terdekat dengan tempat Anda berencana untuk menjalankan kode Anda. Untuk daftar kode wilayah yang akan digunakan dalam `config` file, lihat Referensi Umum Amazon Web Services 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 untuk diakses SDK.

Langkah-langkah untuk mengatur **`credentials`** file lokal dengan kredensi sementara

1. Buat `credentials` file bersama.
2. Dalam `credentials` file, rekatkan teks placeholder berikut hingga Anda menempelkan kredensi 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 di 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 kredenal manual untuk menyalin kredensi peran IAM dari portal akses. AWS

   a. Untuk langkah 4 dalam petunjuk terkait, pilih nama peran IAM yang memberikan akses untuk kebutuhan pengembangan Anda. Peran ini biasanya memiliki nama seperti PowerUserAccessatau Pengembang.

   b. Untuk langkah 7, pilih opsi Tambahkan profil ke file AWS kredensial Anda secara manual dan salin isinya.

6. Rekatkan kredensi yang disalin ke `credentials` file lokal Anda dan hapus nama profil apa pun yang ditempelkan. File Anda harus menyerupai yang 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 kredensi sementara ini ketika membuat klien layanan dan menggunakannya untuk setiap permintaan. Pengaturan untuk peran IAM yang dipilih pada langkah 5a menentukan berapa lama kredensi sementara valid. Durasi maksimum adalah dua belas jam.

Setelah kredensi sementara kedaluwarsa, ulangi langkah 4 hingga 7.

Instal Lingkungan Pengembangan Java

AWS SDK untuk Java V1 membutuhkan Java 7 JDK atau yang lebih baru dan semua Java LTS (dukungan jangka panjang) versi JDK didukung. Jika Anda menggunakan SDK versi 1.12.767 atau lebih lama, Anda dapat menggunakan Java 7, tetapi jika Anda menggunakan SDK versi 1.12.768 atau yang lebih baru, Java 8 diperlukan. Repositori pusat Maven mencantumkan versi terbaru SDK for Java.

Ini AWS SDK untuk Java bekerja dengan Oracle Java SE Development Kit dan dengan distribusi Open Java Development Kit (OpenJDK) seperti Amazon Corretto, Red Hat OpenJDK, dan Adoptium.

Cara untuk mendapatkan AWS SDK untuk Java

Prasyarat

Untuk menggunakan AWS SDK untuk Java, Anda harus memiliki:

- Anda harus dapat masuk ke portal AWS akses yang tersedia di AWS IAM Identity Center.
- Instalasi Java yang cocok.
- Kredensi sementara disiapkan di file bersama `credentials` lokal Anda.

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

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

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

Unduh dan ekstrak SDK (tidak disarankan)

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

> **Note**
>
> Untuk selengkapnya tentang cara mengunduh dan membuat versi SDK sebelumnya, lihat Menginstal SDK versi sebelumnya.

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

SDK berisi direktori berikut:

- `documentation`- berisi dokumentasi API (juga tersedia di web: Referensi AWS SDK untuk Java API).
- `lib`- berisi `.jar` file SDK.
- `samples`- berisi kode contoh kerja yang menunjukkan cara menggunakan SDK.
- `third-party/lib`- berisi pustaka pihak ketiga yang digunakan oleh SDK, seperti Apache commons logging, aspectJ dan framework Spring.

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

Buat versi SDK sebelumnya dari sumber (tidak disarankan)

Hanya versi terbaru dari SDK lengkap yang disediakan dalam bentuk pra-bangun sebagai toples yang dapat diunduh. Namun, Anda dapat membangun versi SDK sebelumnya menggunakan Apache

Maven (open source). Maven akan mengunduh semua dependensi yang diperlukan, membangun dan menginstal SDK dalam satu langkah. Kunjungi http://maven.apache.org/ untuk petunjuk instalasi dan informasi lebih lanjut.

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

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

   `.jar`File yang dihasilkan dibangun ke dalam `target` direktori.
7. (Opsional) Buat dokumentasi Referensi API menggunakan perintah berikut:

   ```
   mvn javadoc:javadoc
   ```

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

Gunakan alat build

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

Topik

- Gunakan SDK dengan Apache Maven
- Menggunakan SDK dengan Gradle

Gunakan SDK dengan Apache Maven

Anda dapat menggunakan Apache Maven untuk mengkonfigurasi dan membangun AWS SDK untuk Java proyek, atau untuk membangun SDK itu sendiri.

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

Buat paket Maven baru

Untuk membuat paket Maven dasar, buka jendela terminal (baris perintah) 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, buat template proyek untuk Anda menggunakan pola dasar quickstart, 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 ke perintah. `archetype:generate` Sebagai contoh:

```
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 mengonfigurasi proyek disediakan di Panduan Memulai Maven.

Konfigurasikan SDK sebagai dependensi Maven

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

Menentukan modul SDK individu

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

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

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

Untuk melihat AWS SDK untuk Java BOM versi terbaru 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 file proyek Anda. `pom.xml`

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

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
```

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

Anda juga dapat merujuk ke Katalog Kode Sampel AWS untuk mempelajari dependensi apa yang akan digunakan untuk diberikan. Layanan AWS Lihat file POM di bawah contoh layanan tertentu. Misalnya, jika Anda tertarik dengan dependensi untuk layanan AWS S3, lihat contoh lengkapnya di. GitHub (Lihat pom under /java/example_code/s 3).

Mengimpor semua modul SDK

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

```
<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 dapat membangunnya menggunakan perintah Maven: `package`

```
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, unduh kode SDK dari GitHub, buka kemasannya secara lokal, lalu jalankan perintah Maven berikut:

```
mvn clean install
```

Menggunakan SDK dengan Gradle

Untuk mengelola dependensi SDK untuk proyek Gradle Anda, impor BOM Maven ke dalam AWS SDK untuk Java file aplikasi. `build.gradle`

> **Note**
>
> Dalam contoh berikut, ganti *`1.12.529`* dalam file build dengan versi yang valid dari file AWS SDK untuk Java. Temukan versi terbaru di repositori pusat Maven.

Penyiapan proyek untuk Gradle 4.6 atau lebih tinggi

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

1. Jika Anda menggunakan Gradle 5.0 atau yang lebih baru, lewati ke langkah 2. Jika tidak, aktifkan fitur IMPROVED_POM_SUPPORT dalam file. `settings.gradle`

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

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

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

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

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

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

```
    implementation 'com.amazonaws:aws-java-sdk-s3'
    ...
}
```

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

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

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

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
  mavenCentral()
}

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

> **Note**
>
> Pada contoh sebelumnya, ganti dependensi Amazon S3 dengan dependensi AWS layanan yang akan Anda gunakan dalam proyek Anda. Modul (dependensi) yang dikelola oleh AWS SDK untuk Java BOM terdaftar di repositori pusat Maven.

Penyiapan proyek untuk versi Gradle lebih awal dari 4.6

Versi Gradle lebih awal dari 4.6 tidak memiliki dukungan BOM asli. Untuk mengelola AWS SDK untuk Java dependensi proyek Anda, gunakan plugin manajemen dependensi Spring untuk Gradle untuk mengimpor Maven BOM untuk SDK.

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

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

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

2. Tambahkan BOM ke bagian DependencyManagement dari file.

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

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

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

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

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

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

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
```

```
  mavenCentral()
}

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

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

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

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

> **Note**
>
> Pada contoh sebelumnya, ganti dependensi Amazon S3 dengan dependensi AWS layanan yang akan Anda gunakan dalam proyek Anda. Modul (dependensi) yang dikelola oleh AWS SDK untuk Java BOM terdaftar di repositori pusat Maven.

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

Menyiapkan kredensyal AWS sementara dan untuk pengembangan Wilayah AWS

Untuk terhubung ke salah satu layanan yang didukung dengan AWS SDK untuk Java, Anda harus memberikan kredensyal AWS sementara. Rantai penyedia AWS SDKs dan CLIs gunakan untuk

mencari kredensyal AWS sementara di sejumlah tempat berbeda, termasuk variabel system/user lingkungan dan file AWS konfigurasi lokal.

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

- Saat menggunakan instans EC2, buat peran IAM dan kemudian berikan akses instans EC2 Anda ke peran tersebut seperti yang ditunjukkan dalam Menggunakan Peran IAM untuk Memberikan Akses ke Sumber Daya aktif. AWS Amazon EC2
- Siapkan AWS kredensyal dalam Eclipse menggunakan file. AWS Toolkit for Eclipse Lihat Menyiapkan AWS Kredensyal di Panduan AWS Toolkit for Eclipse Pengguna untuk informasi selengkapnya.

Konfigurasikan kredensyal sementara

Anda dapat mengonfigurasi kredensyal sementara untuk AWS SDK untuk Java dalam beberapa cara, tetapi berikut adalah pendekatan yang disarankan:

- Tetapkan kredensyal sementara di file profil AWS kredensial di sistem lokal Anda, yang terletak di:
  - `~/.aws/credentials`di Linux, macOS, atau Unix
  - `C:\Users\USERNAME\.aws\credentials` di Windows

  Lihat the section called “Menyiapkan kredensi sementara untuk SDK” di panduan ini untuk petunjuk tentang cara mendapatkan kredensyal sementara Anda.
- Mengatur`AWS_ACCESS_KEY_ID`,`AWS_SECRET_ACCESS_KEY`, dan variabel `AWS_SESSION_TOKEN` lingkungan.

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

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

  Untuk menetapkan variabel ini di Windows, gunakan :

```
set AWS_ACCESS_KEY_ID=your_access_key_id
```

```
set AWS_SECRET_ACCESS_KEY=your_secret_access_key
set AWS_SESSION_TOKEN=your_session_token
```

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

Setelah Anda menetapkan kredensyal AWS sementara Anda menggunakan salah satu metode ini, mereka akan dimuat secara otomatis oleh dengan menggunakan rantai AWS SDK untuk Java penyedia kredensi default. Untuk informasi lebih lanjut tentang bekerja dengan AWS kredensyal di aplikasi Java Anda, lihat Bekerja dengan AWS Kredensyal.

Menyegarkan kredensyal IMDS

AWS SDK untuk Java Dukungan opt-in menyegarkan kredensyal IMDS 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 memengaruhi ketersediaan yang dirasakan. AWS

```
1. // Refresh credentials using a background thread, automatically every minute. This
will log an error if IMDS is down during
2. // a refresh, but your service calls will continue using the cached credentials
until the credentials are refreshed
3. // again one minute later.
4.
5. InstanceProfileCredentialsProvider credentials =
6.     InstanceProfileCredentialsProvider.createAsyncRefreshingProvider(true);
7.
8. AmazonS3Client.builder()
9.              .withCredentials(credentials)
10.              .build();
11.
12. // This is new: When you are done with the credentials provider, you must close it
to release the background thread.
13. credentials.close();
```

Mengatur Wilayah AWS

Anda harus menetapkan default Wilayah AWS yang akan digunakan untuk mengakses AWS layanan dengan. AWS SDK untuk Java Untuk kinerja jaringan terbaik, pilih wilayah yang secara geografis

dekat dengan Anda (atau dengan pelanggan Anda). Untuk daftar wilayah untuk setiap layanan, lihat Wilayah dan Titik Akhir di Referensi Amazon Web Services Umum.

> **Note**
>
> Jika Anda tidak memilih wilayah, maka us-east-1 akan digunakan secara default.

Anda dapat menggunakan teknik serupa untuk menyetel kredensyal untuk menyetel wilayah default AWS Anda:

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

  File ini harus berisi baris dalam format berikut:

  +

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

  +

  Gantikan yang Anda inginkan Wilayah AWS (misalnya, “us-east-1") untuk your_aws_region.
- Mengatur variabel `AWS_REGION` lingkungan.

  Di Linux, macOS, atau Unix, gunakan:

  ```
  export AWS_REGION=your_aws_region
  ```

  Di Windows, gunakan :

  ```
  set AWS_REGION=your_aws_region
  ```

  Dimana your_aws_region adalah nama yang diinginkan. Wilayah AWS

Menggunakan AWS SDK untuk Java

Bagian ini memberikan informasi umum yang penting tentang pemrograman dengan AWS SDK untuk 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 untuk Java Kode.

Topik

Praktik Terbaik untuk AWS Pengembangan dengan AWS SDK untuk Java

Praktik terbaik berikut dapat membantu Anda menghindari masalah atau masalah saat Anda mengembangkan AWS aplikasi dengan aplikasi AWS SDK untuk 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 untuk 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 untuk 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 diberi nama. AmazonDynamoDBClient Setiap antarmuka layanan memiliki pembangun klien yang sesuai yang dapat Anda gunakan untuk membangun implementasi antarmuka layanan. Class client builder untuk DynamoDB bernama AmazonDynamoDBClientBuilder.

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 untuk 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 kredensi dan. Wilayah AWS Jika kredensyal atau wilayah tidak dapat ditentukan dari lingkungan tempat aplikasi berjalan, panggilan ke `defaultClient` gagal. Lihat Bekerja dengan AWS Kredensil dan Wilayah AWS Seleksi untuk informasi selengkapnya tentang bagaimana kredensil 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 koneksinya 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 untuk Java

Untuk membuat permintaan Amazon Web Services, Anda harus memberikan kredensi AWS sementara AWS SDK untuk 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 untuk Java upaya untuk menemukan kredensyal sementara dengan menggunakan rantai penyedia kredensyal default yang diimplementasikan oleh kelas Default. AWSCredentials ProviderChain Rantai penyedia kredensyal default mencari kredensyal dalam urutan ini:

1. Variabel lingkungan -`AWS_ACCESS_KEY_ID`, `AWS_SECRET_KEY` atau`AWS_SECRET_ACCESS_KEY`, dan`AWS_SESSION_TOKEN`. AWS SDK untuk Java Menggunakan EnvironmentVariableCredentialsProviderkelas untuk memuat kredensyal ini.
2. Properti sistem Java -`aws.accessKeyId`, `aws.secretKey` (tetapi tidak`aws.secretAccessKey`), dan`aws.sessionToken`. AWS SDK untuk Java Kegunaan SystemPropertiesCredentialsProvideruntuk memuat kredensyal ini.
3. Kredensyal Token Identitas Web dari lingkungan atau wadah.
4. File profil kredensi default - biasanya terletak di `~/.aws/credentials` (lokasi dapat bervariasi per platform), dan dibagikan oleh banyak AWS SDKs dan oleh. AWS CLI AWS SDK untuk Java Kegunaan 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 untuk Java Kegunaan 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 untuk Java Kegunaan 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. Untuk informasi selengkapnya, lihat EC2ContainerCredentialsProviderWrapper.

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 kredensyal sementara”
- Untuk mengatur properti sistem Java, lihat tutorial System Properties di situs web resmi Java Tutorial.
- Untuk menyiapkan dan menggunakan kredensil 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 untuk 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 kredensi untuk semua yang didukung secara resmi AWS SDKs dan Alat (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 untuk 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 untuk Java mencari file kredensional 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 berkas kredensial 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 menggunakan antarmuka AWSCredentialsPenyedia sebagai masukan. Contoh berikut menunjukkan cara menggunakan kredensyal lingkungan secara khusus.

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

Untuk daftar lengkap penyedia kredensi AWS SDK untuk Java yang disediakan dan rantai penyedia, lihat Semua Kelas Penerapan yang Dikenal di AWSCredentials Penyedia.

> **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 kredensi default atau penyedia atau rantai penyedia khusus atau khusus tidak berfungsi untuk kode Anda, Anda dapat menyetel kredensional 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 adalah contohnya.

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

Info Selengkapnya

- Mendaftar AWS dan Membuat Pengguna IAM
- Menyiapkan AWS 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. Awalan endpoint setiap layanan didefinisikan dalam antarmuka layanan. Misalnya, awalan DynamoDB endpoint didefinisikan dalam AmazonDynamo DB.

Memilih Wilayah

Dimulai dengan versi 1.4 AWS SDK untuk 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, sebagai gantinya, 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 credential/region penyedia default untuk menentukan wilayah dari lingkungan, gunakan `defaultClient` metode pembuat klien.

```
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 Kredensyal Default) untuk mengatur wilayah untuk pengembangan lokal, dan mengandalkan rantai penyedia wilayah default untuk menentukan wilayah saat berjalan pada AWS infrastruktur. Ini sangat menyederhanakan pembuatan klien dan membuat aplikasi Anda tetap portabel.

Penanganan Pengecualian

Memahami bagaimana dan kapan pengecualian AWS SDK untuk 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 untuk 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 untuk 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 untuk 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 untuk Java Ini menyediakan dua cara: objek masa depan dan metode callback.

Java Berjangka

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

Konfigurasi Kolam Benang

Klien asinkron di AWS SDK untuk Java menyediakan kumpulan utas default yang seharusnya berfungsi untuk sebagian besar aplikasi. Anda dapat menerapkan kustom ExecutorServicedan meneruskannya ke klien AWS SDK untuk 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 untuk Java Panggilan Pencatatan

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

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

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

Berikut adalah contoh output log.

```
2009-12-17 09:53:04,269 [main] DEBUG com.amazonaws.request - Sending
Request: POST https://rds.amazonaws.com / Parameters: (MaxRecords: 20,
Action: DescribeEngineDefaultParameters, SignatureMethod: HmacSHA256,
AWSAccessKeyId: ACCESSKEYID, Version: 2009-10-16, SignatureVersion: 2,
Engine: mysql5.1, Timestamp: 2009-12-17T17:53:04.267Z, Signature:
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 untuk 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 untuk Java

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

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

Berikut adalah contoh output log.

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

Konfigurasi Klien

AWS SDK untuk 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 untuk 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 untuk 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 untuk 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 cache alamat IP untuk jangka waktu tertentu, yang dikenal sebagai (TTL). time-to-live

Karena AWS sumber daya menggunakan entri nama DNS yang terkadang berubah, kami sarankan Anda mengonfigurasi JVM Anda dengan nilai TTL 5 detik. Ini memastikan bahwa ketika alamat IP sumber daya berubah, aplikasi Anda akan dapat menerima dan menggunakan alamat IP baru sumber daya dengan meminta DNS.

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

Cara mengatur JVM TTL

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

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

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

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

Mengaktifkan Metrik untuk AWS SDK untuk Java

AWS SDK untuk Java Dapat menghasilkan metrik untuk visualisasi dan pemantauan dengan Amazon CloudWatch yang mengukur:

- kinerja aplikasi Anda saat mengakses AWS
- kinerja Anda JVMs 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 untuk 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. 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 untuk Java berada di bawah namespace AWSSDK/Java, dan diunggah ke CloudWatch wilayah default (us-east-1). Untuk mengubah wilayah, tentukan dengan menggunakan `cloudwatchRegion` atribut di properti sistem. Misalnya, untuk menyetel CloudWatch wilayah ke us-east-1, gunakan:

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

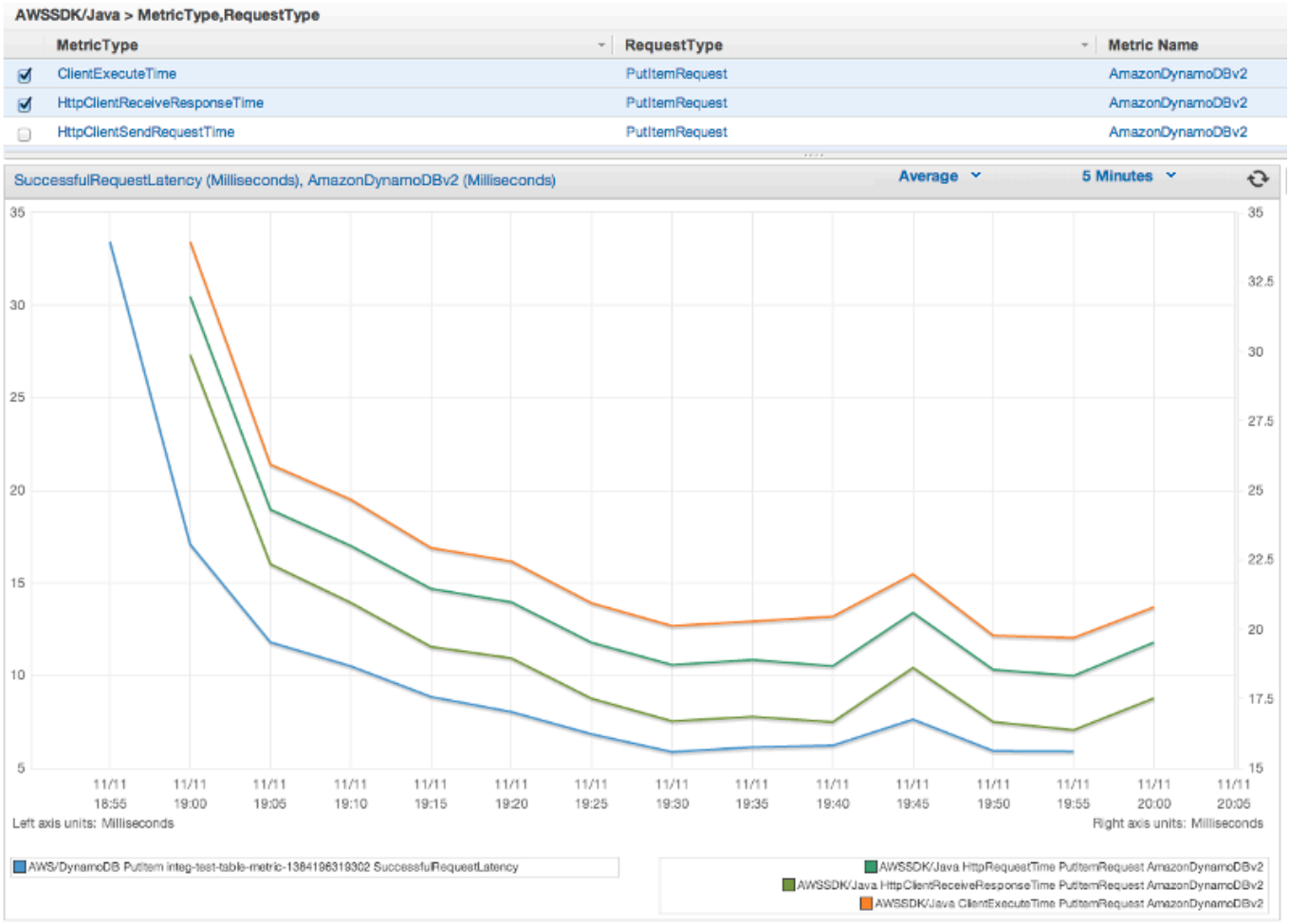
Setelah Anda mengaktifkan fitur, setiap kali ada permintaan layanan AWS dari, titik data metrik akan dibuat AWS SDK untuk Java, antri untuk ringkasan statistik, dan diunggah secara asinkron menjadi sekitar sekali setiap menit. CloudWatch Setelah metrik diunggah, Anda dapat memvisualisasikannya menggunakan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

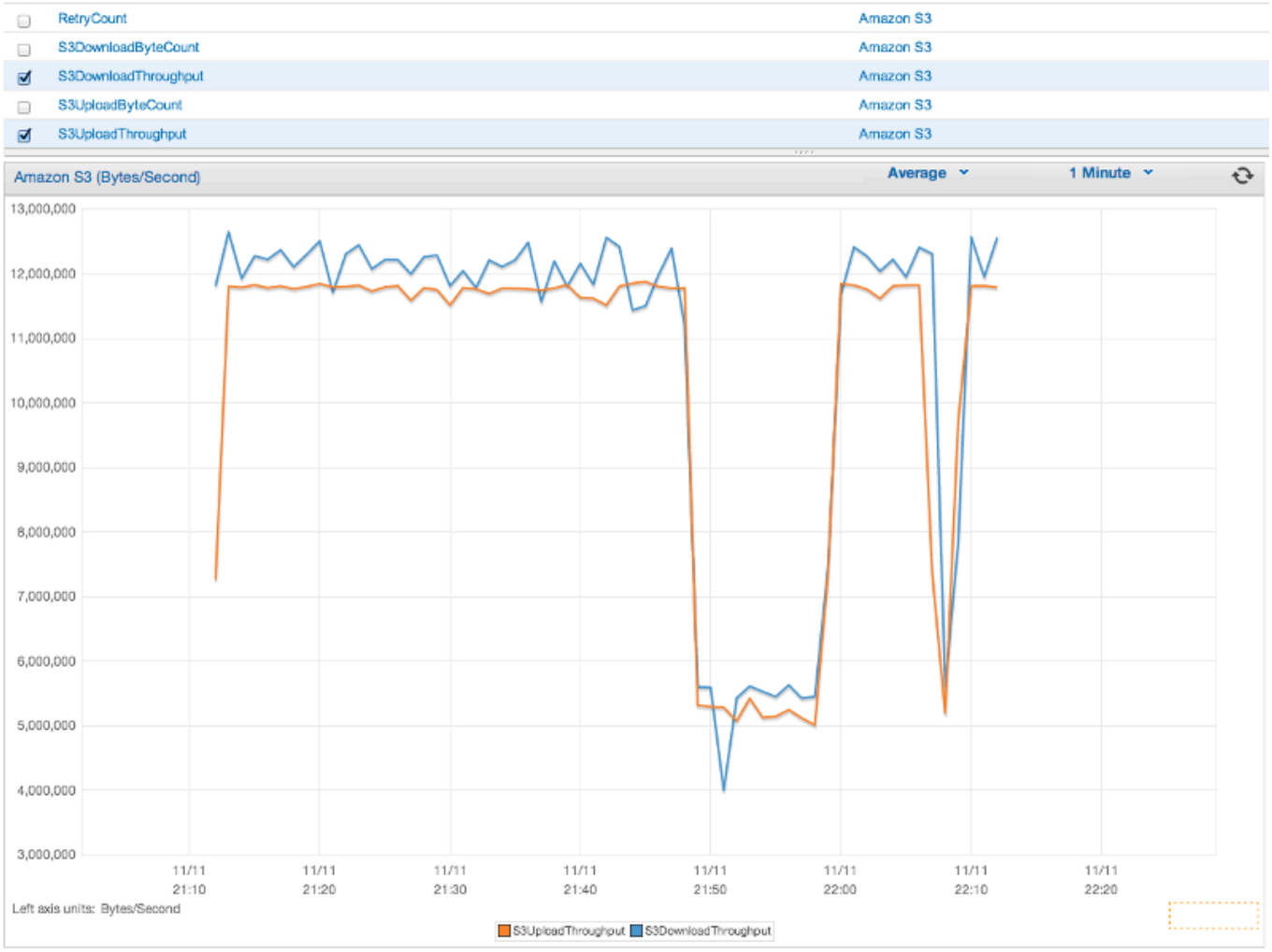
AWS Minta Metrik

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



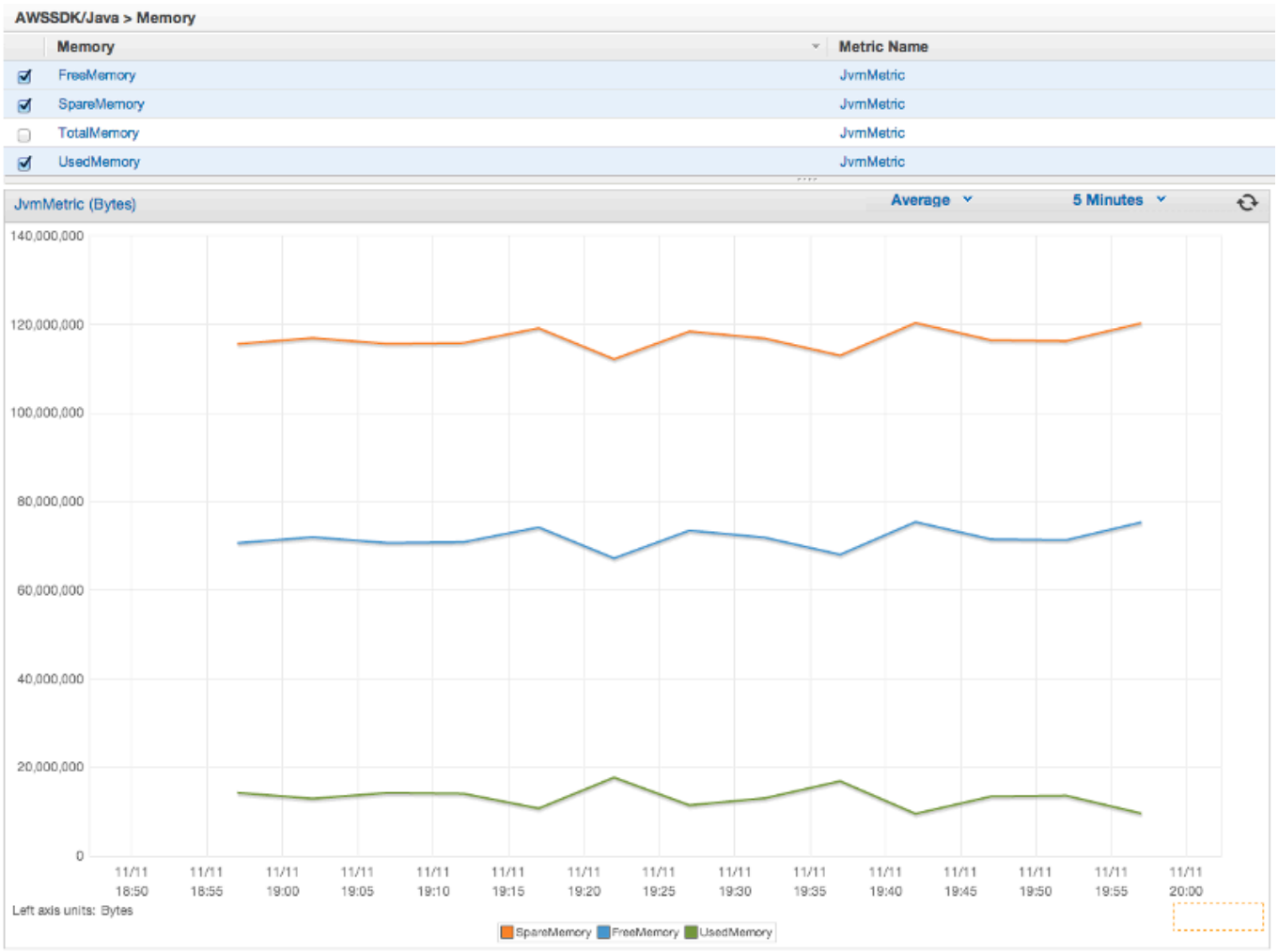
Layanan AWS Metrik

- 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 untuk Java dalam CloudWatch Contoh Menggunakan AWS SDK untuk Java.

- Pelajari lebih lanjut tentang penyetelan kinerja dalam Menyetel AWS SDK untuk Java untuk Meningkatkan Ketahanan posting blog.

AWS SDK untuk Java Contoh Kode

Bagian ini memberikan tutorial dan contoh menggunakan AWS SDK untuk Java v1 untuk AWS layanan program.

Temukan kode sumber untuk contoh-contoh ini dan lainnya dalam repositori contoh kode AWS dokumentasi di. GitHub

Untuk mengusulkan contoh kode baru agar tim AWS dokumentasi mempertimbangkan untuk 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 kontribusi dalam respositori contoh kode di.. GitHub

AWS SDK untuk Java 2.x

Pada 2018, AWS dirilis AWS SDK for Java 2.x. Panduan ini berisi petunjuk tentang penggunaan SDK Java terbaru bersama dengan kode contoh.

> **Note**
>
> Lihat Dokumentasi dan Sumber Daya Tambahan untuk contoh lainnya dan sumber daya tambahan yang tersedia untuk AWS SDK untuk Java pengembang!

CloudWatch Contoh Menggunakan AWS SDK untuk Java

Bagian ini memberikan contoh pemrograman CloudWatchmenggunakan AWS SDK untuk Java.

Amazon CloudWatch memantau sumber daya Amazon Web Services (AWS) Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat menggunakan CloudWatch untuk mengumpulkan dan melacak metrik, yang merupakan variabel yang dapat Anda ukur untuk sumber daya dan aplikasi Anda. CloudWatch alarm mengirim pemberitahuan atau secara otomatis membuat perubahan pada sumber daya yang Anda pantau berdasarkan aturan yang Anda tetapkan.

Untuk informasi selengkapnya CloudWatch, lihat Panduan Amazon CloudWatch Pengguna.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. Kode contoh lengkap tersedia di GitHub. Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

Mendapatkan Metrik dari CloudWatch

Metrik Daftar

Untuk membuat daftar CloudWatch metrik, buat ListMetricsRequestdan panggil `listMetrics` metode AmazonCloudWatchClient ini. Anda dapat menggunakan `ListMetricsRequest` untuk memfilter metrik yang dikembalikan berdasarkan namespace, nama metrik, atau dimensi.

> **Note**
>
> Daftar metrik dan dimensi yang diposting oleh AWS layanan dapat ditemukan dalam {https---docs-aws-amazon-com- AmazonCloudWatch -Latest-Monitoring-CW-Support-for-AWS-html} [Referensi Metrik dan Dimensi Amazon] di Panduan Pengguna. CloudWatch Amazon CloudWatch

Impor

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

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

Kode

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

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

boolean done = false;

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

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

    request.setNextToken(response.getNextToken());

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

Metrik dikembalikan dalam a ListMetricsResultdengan memanggil `getMetrics` metodenya. Hasilnya mungkin paged. Untuk mengambil batch hasil berikutnya, panggil `setNextToken` objek permintaan asli dengan nilai kembali dari `getNextToken` metode `ListMetricsResult` objek, dan meneruskan objek permintaan yang dimodifikasi kembali ke `listMetrics` panggilan lain.

Informasi Selengkapnya

- ListMetricsdi Referensi Amazon CloudWatch API.

Menerbitkan Data Metrik Kustom

Sejumlah AWS layanan mempublikasikan metrik mereka sendiri di ruang nama yang dimulai dengan "AWS" Anda juga dapat mempublikasikan data metrik khusus menggunakan namespace Anda sendiri (asalkan tidak dimulai dengan "“). AWS

Publikasikan Data Metrik Kustom

Untuk mempublikasikan data metrik Anda sendiri, panggil `putMetricData` metode ini dengan file PutMetricDataRequest. AmazonCloudWatchClient `PutMetricDataRequest`Harus menyertakan namespace khusus yang akan digunakan untuk data, dan informasi tentang titik data itu sendiri dalam suatu MetricDatumobjek.

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

Impor

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

Kode

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

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

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

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

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

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

Informasi Selengkapnya

- Menggunakan Amazon CloudWatch Metrik dalam Panduan Amazon CloudWatch Pengguna.
- AWS Ruang nama di Amazon CloudWatch Panduan Pengguna.
- PutMetricDatadalam Referensi Amazon CloudWatch API.

Bekerja dengan CloudWatch Alarm

Buat Alarm

Untuk membuat alarm berdasarkan CloudWatch metrik, panggil `putMetricAlarm` metode dengan PutMetricAlarmRequestdiisi dengan kondisi alarm. AmazonCloudWatchClient

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 membuat daftar CloudWatch alarm yang telah Anda buat, panggil `describeAlarms` metode ini dengan DescribeAlarmsRequestyang dapat Anda gunakan untuk mengatur opsi untuk hasilnya. AmazonCloudWatchClient

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 memanggil `getMetricAlarms` DescribeAlarmsResultyang dikembalikan olehdescribeAlarms.

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

> **Note**
>
> Anda juga dapat mengambil alarm untuk metrik tertentu dengan menggunakan metode ini AmazonCloudWatchClient. `describeAlarmsForMetric` Penggunaannya mirip dengandescribeAlarms.

Hapus Alarm

Untuk menghapus CloudWatch alarm, panggil `deleteAlarms` metode ini dengan DeleteAlarmsRequestberisi satu atau beberapa nama alarm yang ingin Anda hapus. AmazonCloudWatchClient

Impor

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

Kode

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

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

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

Informasi Selengkapnya

- Membuat Amazon CloudWatch Alarm di Amazon CloudWatch Panduan Pengguna
- PutMetricAlarmdi Referensi Amazon CloudWatch API
- DescribeAlarmsdi Referensi Amazon CloudWatch API
- DeleteAlarmsdi Referensi Amazon CloudWatch API

Menggunakan Tindakan Alarm di CloudWatch

Dengan menggunakan tindakan CloudWatch alarm, Anda dapat membuat alarm yang melakukan tindakan seperti menghentikan, menghentikan, me-reboot, atau memulihkan instans secara otomatis. Amazon EC2

> **Note**
>
> Tindakan alarm dapat ditambahkan ke alarm dengan menggunakan `setAlarmActions` metode ini saat membuat alarm. PutMetricAlarmRequest

Aktifkan Tindakan Alarm

Untuk mengaktifkan tindakan alarm untuk CloudWatch alarm, panggil AmazonCloudWatchClient's `enableAlarmActions` dengan EnableAlarmActionsRequestberisi 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);
```

Nonaktifkan Tindakan Alarm

Untuk menonaktifkan tindakan alarm untuk CloudWatch alarm, hubungi AmazonCloudWatchClient's `disableAlarmActions` dengan DisableAlarmActionsRequestberisi satu atau beberapa 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, Menghentikan, Memulai Ulang, atau Memulihkan Instance di Panduan Pengguna Amazon CloudWatch

- PutMetricAlarmdi Referensi Amazon CloudWatch API
- EnableAlarmActionsdi Referensi Amazon CloudWatch API
- DisableAlarmActionsdi Referensi Amazon CloudWatch API

Mengirim Acara ke CloudWatch

CloudWatch Peristiwa memberikan aliran peristiwa sistem yang mendekati real-time yang menggambarkan perubahan AWS sumber daya ke Amazon EC2 instance, Lambda fungsi, Kinesis aliran, Amazon ECS tugas, mesin Step Functions status, Amazon SNS topik, Amazon SQS antrian, atau target bawaan. Anda dapat mencocokkan acara dan meruteknya ke satu atau beberapa fungsi atau aliran target dengan menggunakan aturan sederhana.

Tambahkan Acara

Untuk menambahkan CloudWatch peristiwa khusus, panggil `putEvents` metode dengan PutEventsRequestobjek yang berisi satu atau beberapa PutEventsRequestEntryobjek yang memberikan detail tentang setiap peristiwa. AmazonCloudWatchEventsClient Anda dapat menentukan beberapa parameter untuk entri seperti sumber dan jenis acara, sumber daya yang terkait dengan acara, dan sebagainya.

> **Note**
>
> Anda dapat menentukan maksimum 10 acara per panggilan ke`putEvents`.

Impor

```
import 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 `putRule` metode PutRuleRequestdengan nama aturan dan parameter opsional seperti pola acara, IAM peran untuk dikaitkan dengan aturan, dan ekspresi penjadwalan yang menjelaskan seberapa sering aturan dijalankan. AmazonCloudWatchEventsClient

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);
```

Tambahkan Target

Target adalah sumber daya yang dipanggil ketika suatu aturan dipicu. Contoh target termasuk Amazon EC2 instance, Lambda fungsi, Kinesis aliran, Amazon ECS tugas, mesin Step Functions status, dan target bawaan.

Untuk menambahkan target ke aturan, panggil `putTargets` metode dengan PutTargetsRequestberisi aturan untuk diperbarui dan daftar target untuk ditambahkan ke aturan. AmazonCloudWatchEventsClient

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 Acara dengan PutEvents di Panduan Amazon CloudWatch Events Pengguna
- Jadwalkan Ekspresi untuk Aturan di Panduan Amazon CloudWatch Events Pengguna
- Jenis Acara untuk CloudWatch Acara di Panduan Amazon CloudWatch Events Pengguna
- Peristiwa dan Pola Peristiwa di Panduan Amazon CloudWatch Events Pengguna
- PutEventsdi Referensi Amazon CloudWatch Events API

- PutTargetsdi Referensi Amazon CloudWatch Events API
- PutRuledi Referensi Amazon CloudWatch Events API

DynamoDB Contoh Menggunakan AWS SDK untuk Java

Bagian ini memberikan contoh pemrograman DynamoDBmenggunakan AWS SDK untuk Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. Kode contoh lengkap tersedia di GitHub. Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

- Gunakan AWS titik akhir berbasis akun
- Bekerja dengan Tabel di DynamoDB
- Bekerja dengan Item di DynamoDB

Gunakan AWS titik akhir berbasis akun

DynamoDB AWS menawarkan endpoint berbasis akun yang dapat meningkatkan kinerja dengan menggunakan ID akun AWS Anda untuk merampingkan perutean permintaan.

Untuk memanfaatkan fitur ini, Anda perlu menggunakan versi 1.12.771 atau lebih tinggi dari versi 1. AWS SDK untuk Java Anda dapat menemukan versi terbaru SDK yang tercantum di repositori pusat Maven. Setelah versi SDK yang didukung aktif, SDK secara otomatis menggunakan titik akhir baru.

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

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

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

```
ClientConfiguration config = new ClientConfiguration()
    .withAccountIdEndpointMode(AccountIdEndpointMode.DISABLED);
AWSCredentialsProvider credentialsProvider = new
 EnvironmentVariableCredentialsProvider();

AmazonDynamoDB dynamodb = AmazonDynamoDBClientBuilder.standard()
    .withClientConfiguration(config)
    .withCredentials(credentialsProvider)
    .withRegion(Regions.US_WEST_2)
    .build();
```

Panduan Referensi AWS SDKs and Tools memberikan informasi lebih lanjut tentang tiga opsi konfigurasi terakhir.

Bekerja dengan Tabel di DynamoDB

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

Untuk setiap tabel, Anda harus mendefinisikan:

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

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

  Setiap nilai kunci memiliki tipe data terkait, disebutkan oleh kelas. ScalarAttributeType Nilai kunci dapat berupa biner (B), numerik (N), atau string (S). Untuk informasi selengkapnya, lihat Aturan Penamaan dan Jenis Data di Panduan Amazon DynamoDB Pengembang.
- Nilai throughput yang disediakan yang menentukan jumlah unit kapasitas baca/tulis yang dicadangkan untuk tabel.

> **Note**
>
> Amazon DynamoDB Penetapan harga didasarkan pada nilai throughput yang disediakan yang Anda tetapkan pada tabel Anda, jadi cadangkan hanya kapasitas sebanyak yang Anda pikir Anda perlukan untuk tabel Anda.

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

Buat Tabel

Gunakan `createTable` metode DynamoDB klien untuk membuat DynamoDB tabel baru. Anda perlu membangun atribut tabel dan skema tabel, yang keduanya digunakan untuk mengidentifikasi kunci utama tabel Anda. Anda juga harus menyediakan nilai throughput awal yang disediakan dan nama tabel. Hanya tentukan atribut tabel kunci saat membuat DynamoDB 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 Primer Sederhana

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

Kode

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

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

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

Lihat contoh lengkapnya di GitHub.

Membuat Tabel dengan Composite Primary Key

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 lengkapnya di GitHub.

Daftar Tabel

Anda dapat membuat daftar tabel di wilayah tertentu dengan memanggil `listTables` metode DynamoDB klien.

> **Note**
>
> Jika tabel bernama tidak ada untuk akun dan wilayah Anda, a 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 lengkapnya di GitHub.

Jelaskan (Dapatkan Informasi tentang) Tabel

Panggil `describeTable` metode DynamoDB klien.

> **Note**
>
> Jika tabel bernama tidak ada untuk akun dan wilayah Anda, a 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 lengkapnya di GitHub.

Ubah (Perbarui) Tabel

Anda dapat memodifikasi nilai throughput yang disediakan tabel kapan saja dengan memanggil metode DynamoDB klien. `updateTable`

> **Note**
>
> Jika tabel bernama tidak ada untuk akun dan wilayah Anda, a 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 lengkapnya di GitHub.

Menghapus Tabel

Panggil `deleteTable` metode DynamoDB klien dan berikan nama tabel.

> **Note**
>
> Jika tabel bernama tidak ada untuk akun dan wilayah Anda, a 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 lengkapnya di GitHub.

Info Selengkapnya

- Pedoman untuk Bekerja dengan Tabel di Panduan Amazon DynamoDB Pengembang
- Bekerja dengan Tabel DynamoDB di Panduan Amazon DynamoDB Pengembang

Bekerja dengan Item di DynamoDB

Dalam DynamoDB, item adalah kumpulan atribut, yang masing-masing memiliki nama dan nilai. Nilai atribut dapat berupa skalar, set, atau jenis dokumen. Untuk informasi selengkapnya, lihat Aturan Penamaan dan Jenis Data di Panduan Amazon DynamoDB Pengembang.

Mengambil (Dapatkan) Item dari Tabel

Panggil `getItem` metode AmazonDynamo DB dan berikan GetItemRequestobjek dengan nama tabel dan nilai kunci primer dari item yang Anda inginkan. Ia mengembalikan sebuah GetItemResultobjek.

Anda dapat menggunakan `getItem()` metode `GetItemResult` objek yang dikembalikan untuk mengambil Map of key (String) dan value (AttributeValue) pasangan yang terkait dengan item tersebut.

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);
```

Lihat contoh lengkapnya di GitHub.

Menambahkan Item Baru ke Tabel

Buat Peta pasangan kunci-nilai yang mewakili atribut item. Ini harus menyertakan nilai untuk bidang kunci utama tabel. Jika item yang diidentifikasi oleh kunci utama sudah ada, bidangnya diperbarui oleh permintaan.

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

Impor

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

Kode

```
HashMap<String,AttributeValue> item_values =
```

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

item_values.put("Name", new AttributeValue(name));

for (String[] field : extra_fields) {
    item_values.put(field[0], new AttributeValue(field[1]));
}

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

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

Lihat contoh lengkapnya di GitHub.

Memperbarui Item yang Ada dalam Tabel

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

> **Note**
>
> Jika tabel bernama tidak ada untuk akun dan wilayah Anda, atau jika item yang diidentifikasi oleh kunci utama yang Anda lewati tidak ada, a akan ResourceNotFoundExceptionditampilkan.

Impor

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

```
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.AttributeValueUpdate;
import com.amazonaws.services.dynamodbv2.model.ResourceNotFoundException;
import java.util.ArrayList;
```

Kode

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

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

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

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

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

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

Lihat contoh lengkapnya di GitHub.

Gunakan kelas Dynamo DBMapper

AWS SDK untuk JavaIni menyediakan DBMapper kelas Dynamo, memungkinkan Anda untuk memetakan kelas sisi klien Anda ke tabel. Amazon DynamoDB Untuk menggunakan DBMapper kelas Dynamo, Anda menentukan hubungan antara item dalam DynamoDB tabel dan instance objek yang sesuai dalam kode Anda dengan menggunakan anotasi (seperti yang ditunjukkan dalam contoh kode berikut). DBMapperKelas Dynamo memungkinkan Anda untuk mengakses tabel Anda; melakukan berbagai operasi membuat, membaca, memperbarui, dan menghapus (CRUD); dan mengeksekusi query.

> **Note**
>
> DBMapperKelas Dynamo tidak memungkinkan Anda untuk 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 cara menambahkan konten ke tabel Musik dengan menggunakan DBMapper kelas Dynamo. Setelah konten ditambahkan ke tabel, perhatikan bahwa item dimuat dengan menggunakan tombol Partition and Sort. Kemudian item Penghargaan diperbarui. Untuk informasi tentang cara membuat tabel Musik, lihat Membuat Tabel di Panduan Amazon DynamoDB Pengembang.

```
        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;
        }
    }
```

Lihat contoh lengkapnya di GitHub.

Info Selengkapnya

- Pedoman untuk Bekerja dengan Item di Panduan Amazon DynamoDB Pengembang
- Bekerja dengan Item DynamoDB di Panduan Amazon DynamoDB Pengembang

Amazon EC2 Contoh Menggunakan AWS SDK untuk Java

Bagian ini memberikan contoh pemrograman Amazon EC2dengan AWS SDK untuk Java.

Topik

- Tutorial: Memulai Sebuah EC2 Instance
- Menggunakan Peran IAM untuk Memberikan Akses ke AWS Sumber Daya Amazon EC2
- Tutorial: Contoh Amazon EC2 Spot
- Tutorial: Manajemen Permintaan Amazon EC2 Spot Tingkat Lanjut
- Mengelola Amazon EC2 Instans

- Menggunakan Alamat IP Elastis di Amazon EC2
- Gunakan wilayah dan zona ketersediaan
- Bekerja dengan Pasangan Amazon EC2 Kunci
- Bekerja dengan kelompok keamanan di Amazon EC2

Tutorial: Memulai Sebuah EC2 Instance

Tutorial ini menunjukkan bagaimana menggunakan AWS SDK untuk Java untuk memulai sebuah EC2 instance.

Topik

- Prasyarat
- Buat Grup Amazon EC2 Keamanan
- Membuat Pasangan Kunci
- Jalankan sebuah Amazon EC2 Instance

Prasyarat

Sebelum Anda mulai, pastikan bahwa Anda telah membuat Akun AWS dan bahwa Anda telah mengatur AWS kredensi Anda. Untuk informasi selengkapnya, lihat Memulai.

Buat Grup Amazon EC2 Keamanan

EC2-Classic pensiun

> **Warning**
>
> Kami pensiun EC2 -Classic pada 15 Agustus 2022. Kami menyarankan Anda bermigrasi dari EC2 -Classic ke VPC. Untuk informasi lebih lanjut, lihat posting blog EC2-Classic-Classic Networking is Retiring - Inilah Cara Mempersiapkan.

Buat grup keamanan, yang bertindak sebagai firewall virtual yang mengontrol lalu lintas jaringan untuk satu atau lebih EC2 contoh. Secara default, Amazon EC2 kaitkan instans Anda dengan grup keamanan yang tidak mengizinkan lalu lintas masuk. Anda dapat membuat grup keamanan yang memungkinkan EC2 instans Anda menerima lalu lintas tertentu. Misalnya, jika Anda perlu terhubung

ke instance Linux, Anda harus mengkonfigurasi grup keamanan untuk mengizinkan lalu lintas SSH. Anda dapat membuat grup keamanan menggunakan Amazon EC2 konsol atau file AWS SDK untuk Java.

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

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

1. Membuat dan menginisialisasi sebuah CreateSecurityGroupRequestinstance. Gunakan withGroupNamemetode untuk mengatur nama grup keamanan, dan metode withDescription untuk mengatur deskripsi grup keamanan, sebagai berikut:

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

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

2. Lulus objek permintaan sebagai parameter ke createSecurityGroupmetode. Metode mengembalikan CreateSecurityGroupResultobjek, sebagai berikut:

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

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

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

1. Membuat dan menginisialisasi sebuah IpPermissioninstance. Gunakan metode withIPv4Ranges untuk mengatur rentang alamat IP untuk mengotorisasi masuknya, dan gunakan withIpProtocolmetode untuk mengatur protokol IP. Gunakan withToPortmetode withFromPortdan untuk menentukan rentang port untuk mengotorisasi masuknya, sebagai berikut:

```
IpPermission ipPermission =
    new IpPermission();

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

ipPermission.withIpv4Ranges(Arrays.asList(new IpRange[] {ipRange1, ipRange2}))
            .withIpProtocol("tcp")
            .withFromPort(22)
            .withToPort(22);
```

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

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

2. Membuat dan menginisialisasi sebuah AuthorizeSecurityGroupIngressRequestinstance. Gunakan `withGroupName` metode untuk menentukan nama grup keamanan, dan teruskan `IpPermission` objek yang Anda inisialisasi sebelumnya ke withIpPermissionsmetode, sebagai berikut:

```
AuthorizeSecurityGroupIngressRequest authorizeSecurityGroupIngressRequest =
    new AuthorizeSecurityGroupIngressRequest();

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

3. Lulus objek permintaan ke dalam metode authorizeSecurityGroupIngress, sebagai berikut:

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

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

Setiap kali Anda memanggil metode authorizeSecurityGroupIngress atau authorizeSecurityGroupEgress, aturan ditambahkan ke grup keamanan Anda.

Membuat Pasangan Kunci

Anda harus menentukan key pair ketika Anda meluncurkan sebuah EC2 instance dan kemudian menentukan kunci pribadi dari key pair ketika Anda terhubung ke instance. Anda dapat membuat key pair atau menggunakan key pair yang sudah ada yang telah Anda gunakan saat meluncurkan instance lain. Untuk informasi selengkapnya, lihat Pasangan Amazon EC2 Kunci di Panduan Amazon EC2 Pengguna untuk Instans Linux.

1. Membuat dan menginisialisasi sebuah CreateKeyPairRequestinstance. Gunakan withKeyNamemetode untuk mengatur nama key pair, sebagai berikut:

```
CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();

createKeyPairRequest.withKeyName(keyName);
```

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

2. Lulus objek permintaan ke createKeyPairmetode. Metode mengembalikan sebuah CreateKeyPairResultcontoh, sebagai berikut:

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

3. Panggil getKeyPairmetode objek hasil untuk mendapatkan KeyPairobjek. Panggil getKeyMaterialmetode `KeyPair` objek untuk mendapatkan kunci pribadi yang dikodekan PEM yang tidak terenkripsi, sebagai berikut:

```
KeyPair keyPair = new KeyPair();

keyPair = createKeyPairResult.getKeyPair();

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

Jalankan sebuah Amazon EC2 Instance

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

1. Membuat dan menginisialisasi sebuah RunInstancesRequestinstance. Pastikan bahwa AMI, key pair, dan grup keamanan yang Anda tentukan ada di wilayah yang Anda tentukan saat Anda 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 publik yang AMIs disediakan oleh Amazon atau membuat milik Anda sendiri, lihat Amazon Machine Image (AMI).

withInstanceType

- Jenis instance yang kompatibel dengan AMI yang ditentukan. Untuk informasi selengkapnya, lihat Jenis Instance dalam Panduan Amazon EC2 Pengguna untuk Instans Linux.

withMinCount

- Jumlah minimum EC2 instance yang akan diluncurkan. Jika ini lebih banyak instance daripada yang Amazon EC2 dapat diluncurkan di Zona Ketersediaan target, tidak akan Amazon EC2 meluncurkan instance.

withMaxCount

- Jumlah maksimum EC2 instance untuk diluncurkan. Jika ini lebih banyak instance daripada yang Amazon EC2 dapat diluncurkan di Zona Ketersediaan target, Amazon EC2 meluncurkan jumlah instans sebanyak mungkin di atas. `MinCount` Anda dapat meluncurkan antara 1 dan jumlah maksimum instance yang diizinkan untuk jenis instans. Untuk informasi

selengkapnya, lihat Berapa banyak instance yang dapat saya jalankan Amazon EC2 di FAQ Amazon EC2 Umum.

withKeyName

- Nama dari EC2 key pair. Jika Anda meluncurkan instance tanpa menentukan key pair, Anda tidak dapat menghubungkannya. Untuk informasi selengkapnya, lihat Membuat Pasangan Kunci.

withSecurityGroups

- Satu atau lebih grup keamanan. Untuk informasi selengkapnya, lihat Membuat Grup Amazon EC2 Keamanan.

2. Luncurkan instance dengan meneruskan objek permintaan ke metode RunInstances. Metode mengembalikan RunInstancesResultobjek, sebagai berikut:

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

Setelah instans Anda berjalan, Anda dapat menghubungkannya menggunakan key pair Anda. Untuk informasi selengkapnya, lihat Connect to Your Linux Instance. di Panduan Amazon EC2 Pengguna untuk Instans Linux.

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

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

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

Rantai penyedia default dan profil EC2 instans

Jika aplikasi Anda membuat AWS klien menggunakan konstruktor default, maka klien akan mencari kredensil menggunakan rantai penyedia kredensi default, dengan 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 kredensi default (lokasi file ini bervariasi menurut platform).
4. Kredensil dikirimkan melalui layanan Amazon EC2 kontainer jika variabel `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` lingkungan disetel dan manajer keamanan memiliki izin untuk mengakses variabel.
5. Dalam kredensi profil instance, yang ada dalam metadata instance yang terkait dengan peran IAM untuk instance. EC2
6. Kredensi Token Identitas Web dari lingkungan atau wadah.

Langkah kredensial profil instans dalam rantai penyedia default hanya tersedia saat menjalankan aplikasi Anda pada sebuah Amazon EC2 instance, tetapi memberikan kemudahan penggunaan dan keamanan terbaik saat bekerja dengan Amazon EC2 instance. Anda juga dapat meneruskan InstanceProfileCredentialsProviderinstance langsung ke konstruktor klien untuk mendapatkan kredensil profil instance tanpa melanjutkan seluruh rantai penyedia default.

Sebagai contoh:

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

Saat menggunakan pendekatan ini, SDK mengambil AWS kredensil sementara yang memiliki izin yang sama dengan yang terkait dengan peran IAM yang terkait dengan instance di profil instancenya. Amazon EC2 Meskipun kredensil ini bersifat sementara dan pada akhirnya akan kedaluwarsa, `InstanceProfileCredentialsProvider` secara berkala menyegarkannya untuk Anda sehingga kredensil yang diperoleh terus memungkinkan akses ke. AWS

> **Important**
>
> Penyegaran kredenal otomatis hanya terjadi ketika Anda menggunakan konstruktor klien default, yang membuatnya sendiri `InstanceProfileCredentialsProvider` sebagai bagian dari rantai penyedia default, atau ketika Anda meneruskan `InstanceProfileCredentialsProvider` instance langsung ke konstruktor klien. Jika Anda menggunakan metode lain untuk mendapatkan atau meneruskan kredensi profil instans, Anda bertanggung jawab untuk memeriksa dan menyegarkan kredensil yang kedaluwarsa.

Jika konstruktor klien tidak dapat menemukan kredensil menggunakan rantai penyedia kredensil, itu akan memunculkan file. AmazonClientException

Walkthrough: Menggunakan peran IAM untuk instance EC2

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

Buat IAM Role

Buat peran IAM yang memberikan akses hanya-baca ke. Amazon S3

1. Buka konsol IAM.
2. Di panel navigasi, pilih Peran, lalu Buat Peran Baru.
3. Masukkan nama untuk peran tersebut, lalu pilih Langkah Berikutnya. Ingat nama ini, karena Anda akan membutuhkannya saat meluncurkan Amazon EC2 instance Anda.
4. Pada halaman Pilih Jenis Peran, di bawah Layanan AWS Peran, pilih Amazon EC2 .
5. Pada halaman Setel Izin, di bawah Pilih Templat Kebijakan, pilih Akses Hanya Amazon S3 Baca, lalu Langkah Berikutnya.
6. Pada halaman Ulasan, pilih Buat Peran.

Luncurkan EC2 Instance dan Tentukan Peran IAM Anda

Anda dapat meluncurkan Amazon EC2 instance dengan peran IAM menggunakan Amazon EC2 konsol atau. AWS SDK untuk Java

- Untuk meluncurkan Amazon EC2 instance menggunakan konsol, ikuti petunjuk di Memulai dengan Instans Amazon EC2 Linux di Panduan Amazon EC2 Pengguna untuk Instans Linux.

  Saat Anda mencapai halaman Peluncuran Instance Tinjauan, pilih Edit detail instans. Dalam peran IAM, pilih peran IAM yang Anda buat sebelumnya. Selesaikan prosedur sesuai petunjuk.

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

- Untuk meluncurkan Amazon EC2 instance dengan peran IAM menggunakan AWS SDK untuk Java, lihat Menjalankan Amazon EC2 Instance.

Buat Aplikasi Anda

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

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

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

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

```
import java.io.*;

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

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

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

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

```
  }

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

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

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

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

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

Bangun dan jalankan program yang dimodifikasi. Perhatikan bahwa tidak ada kredensil yang disimpan dalam program. Oleh karena itu, kecuali Anda memiliki AWS kredensil Anda sudah ditentukan, kode akan dibuang. AmazonServiceException Sebagai contoh:

```
$ 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 Kompilasi ke EC2 Instance Anda

Transfer program ke Amazon EC2 instans Anda menggunakan salinan aman (), bersama dengan AWS SDK untuk Java pustaka. Urutan perintah terlihat seperti berikut ini.

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

> **Note**
>
> Tergantung pada distribusi Linux yang Anda gunakan, nama pengguna mungkin "ec2-user", "root", atau "ubuntu". Untuk mendapatkan nama DNS publik instance Anda, buka EC2 konsol dan cari nilai DNS Publik di tab Deskripsi (misalnya,`ec2-198-51-100-1.compute-1.amazonaws.com`).

Dalam perintah sebelumnya:

- `GetS3Object.class`adalah program yang dikompilasi
- `build.xml`adalah file semut yang digunakan untuk membangun dan menjalankan program Anda
- `third-party`direktori `lib` dan adalah folder perpustakaan yang sesuai dari file. AWS SDK untuk Java
- `-r`Sakelar menunjukkan bahwa `scp` harus melakukan salinan rekursif dari semua isi `library` dan `third-party` direktori dalam distribusi. AWS SDK untuk Java

- `-p`Sakelar menunjukkan bahwa `scp` harus mempertahankan izin file sumber saat menyalinnya ke tujuan.

> **Note**
>
> `-p`Sakelar hanya berfungsi di Linux, macOS, atau Unix. Jika Anda menyalin file dari Windows, Anda mungkin perlu memperbaiki izin file pada instance Anda menggunakan perintah berikut:

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

Jalankan Program Sampel pada EC2 Instance

Untuk menjalankan program, sambungkan ke Amazon EC2 instans Anda. Untuk informasi selengkapnya, lihat Connect to Your Linux Instance di Panduan Amazon EC2 Pengguna untuk Instans Linux.

Jika tidak **`ant`** 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 Amazon S3 objek Anda ke jendela perintah Anda.

Tutorial: Contoh Amazon EC2 Spot

Gambaran Umum

Instans Spot memungkinkan Anda menawar kapasitas Amazon Elastic Compute Cloud (Amazon EC2) yang tidak terpakai hingga 90% dibandingkan harga Instans Sesuai Permintaan dan menjalankan instans yang diperoleh selama tawaran Anda melebihi Harga Spot saat ini. Amazon EC2 mengubah Harga Spot secara berkala berdasarkan penawaran dan permintaan, dan pelanggan yang penawarannya memenuhi atau melampauinya mendapatkan akses ke Instans Spot yang

tersedia. Seperti Instans Sesuai Permintaan dan Instans Cadangan, Instans Spot memberi Anda opsi lain untuk mendapatkan lebih banyak kapasitas komputasi.

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

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

Penting untuk dicatat:

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

Instans Spot bekerja persis seperti Amazon EC2 instans lain saat berjalan, dan seperti Amazon EC2 instance lainnya, Instans Spot dapat dihentikan saat Anda tidak lagi membutuhkannya. Jika Anda menghentikan instans, Anda membayar sebagian jam yang digunakan (seperti yang Anda lakukan untuk Instans Sesuai Permintaan atau Cadangan). Namun, jika Harga Spot melebihi tawaran Anda dan instans Anda dihentikan oleh Amazon EC2, Anda tidak akan dikenakan biaya untuk sebagian jam penggunaan.

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

- Kirim Permintaan Spot
- Tentukan kapan Permintaan Spot terpenuhi
- Batalkan Permintaan Spot
- Mengakhiri instance terkait

Prasyarat

Untuk menggunakan tutorial ini, Anda harus AWS SDK untuk Java menginstal, serta telah memenuhi prasyarat instalasi dasarnya. Lihat Mengatur AWS SDK untuk Java untuk informasi lebih lanjut.

Langkah 1: Menyiapkan Kredensi Anda

Untuk mulai menggunakan contoh kode ini, Anda perlu mengatur AWS kredensional. Lihat Menyiapkan AWS Kredensial dan Wilayah untuk Pengembangan untuk petunjuk tentang cara melakukannya.

> **Note**
>
> Kami menyarankan Anda menggunakan kredensi pengguna IAM untuk memberikan nilai-nilai ini. Untuk informasi selengkapnya, lihat Mendaftar AWS dan Membuat Pengguna IAM.

Sekarang setelah Anda mengonfigurasi pengaturan Anda, Anda dapat mulai menggunakan kode dalam contoh.

Langkah 2: Menyiapkan Grup Keamanan

Grup keamanan bertindak sebagai firewall yang mengontrol lalu lintas yang diizinkan masuk dan keluar dari sekelompok instance. Secara default, sebuah instance dimulai tanpa grup keamanan apa pun, yang berarti bahwa semua lalu lintas IP yang masuk, pada port TCP apa pun akan ditolak. Jadi, sebelum mengirimkan Permintaan Spot kami, kami akan membuat grup keamanan yang memungkinkan lalu lintas jaringan yang diperlukan. Untuk keperluan tutorial ini, kami 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 perlu menyertakan atau menjalankan contoh kode berikut yang mengatur grup keamanan secara terprogram.

Setelah kami membuat objek `AmazonEC2` klien, kami membuat `CreateSecurityGroupRequest` objek dengan nama, "GettingStarted" dan deskripsi untuk grup keamanan. Kemudian kita memanggil `ec2.createSecurityGroup` API untuk membuat grup.

Untuk mengaktifkan akses ke grup, kami membuat `ipPermission` objek dengan rentang alamat IP yang 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());
}
```

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

Anda juga dapat membuat grup keamanan menggunakan file AWS Toolkit for Eclipse. Lihat Mengelola Grup Keamanan dari AWS Cost Explorer untuk informasi selengkapnya.

Langkah 3: Mengirimkan Permintaan Spot Anda

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

Ada beberapa jenis contoh untuk dipilih; pergi ke Jenis Amazon EC2 Instance 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 Linux up-to-date Amazon paling banyak tersedia saat kami menulis tutorial ini. AMI terbaru dapat berubah seiring waktu, tetapi Anda selalu dapat menentukan AMI versi terbaru dengan mengikuti langkah-langkah berikut:

1. Buka konsol Amazon EC2.
2. Pilih tombol Launch Instance.
3. Jendela pertama menampilkan yang AMIs tersedia. ID AMI tercantum di sebelah setiap judul AMI. Atau, Anda dapat menggunakan `DescribeImages` API, tetapi memanfaatkan perintah itu berada di luar cakupan tutorial ini.

Ada banyak cara untuk mendekati penawaran untuk Instans Spot; untuk mendapatkan gambaran luas tentang berbagai pendekatan, Anda harus melihat video Penawaran untuk Instans Spot. 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 Sesuai Permintaan Anda memiliki pekerjaan pemrosesan batch yang akan memakan waktu beberapa jam atau hari untuk dijalankan. Namun, Anda fleksibel sehubungan dengan kapan dimulai dan kapan selesai. Anda ingin melihat apakah Anda dapat menyelesaikannya dengan biaya lebih murah dibandingkan dengan Instans Sesuai Permintaan. Anda memeriksa riwayat Harga Spot untuk jenis misalnya menggunakan API Konsol Manajemen AWS atau Amazon EC2 API. Untuk informasi lebih lanjut, buka Melihat Riwayat Harga Spot. Setelah menganalisis riwayat harga untuk jenis instans yang diinginkan 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 On-Demand), mengantisipasi bahwa permintaan Spot satu kali Anda kemungkinan besar akan terpenuhi dan berjalan untuk waktu komputasi yang cukup berturut-turut untuk menyelesaikan pekerjaan.
  - Atau, Anda dapat menentukan jumlah yang bersedia Anda bayarkan untuk Instans Spot sebagai % dari harga Instans Sesuai Permintaan, 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 mengotomatiskan tugas ini nanti dalam tutorial ini.)
- Bayar Tidak Lebih dari Nilai Hasil Anda memiliki pekerjaan pemrosesan data untuk dijalankan. Anda memahami nilai hasil pekerjaan dengan cukup baik untuk mengetahui berapa nilainya dalam hal biaya komputasi. Setelah menganalisis riwayat Harga Spot untuk jenis instans Anda, Anda memilih harga tawaran di mana biaya waktu komputasi tidak lebih dari nilai hasil pekerjaan. Anda membuat tawaran persisten dan membiarkannya berjalan sebentar-sebentar karena Harga Spot berfluktuasi pada atau di bawah tawaran Anda.
- Dapatkan Kapasitas Komputasi dengan Cepat Anda memiliki kebutuhan jangka pendek yang tidak terduga untuk kapasitas tambahan yang tidak tersedia melalui Instans Sesuai Permintaan. Setelah menganalisis riwayat Harga Spot untuk jenis instans Anda, Anda menawar di atas harga historis tertinggi untuk memberikan kemungkinan besar bahwa permintaan Anda akan terpenuhi dengan cepat dan terus menghitung hingga selesai.

Setelah memilih harga bid, Anda siap untuk meminta Instans Spot. Untuk keperluan tutorial ini, kami akan menawar harga On-Demand ($0.03) untuk memaksimalkan peluang bahwa tawaran akan terpenuhi. Anda dapat menentukan jenis instans yang tersedia dan harga Sesuai Permintaan untuk instans dengan membuka halaman Harga. Amazon EC2 Saat Instans Spot berjalan, Anda membayar

harga Spot yang berlaku untuk periode waktu instans Anda berjalan. Harga Instans Spot ditetapkan oleh Amazon EC2 dan disesuaikan 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 Sesuai Permintaan.Untuk meminta Instans Spot, Anda hanya perlu membuat permintaan dengan parameter yang Anda pilih sebelumnya. Kita mulai dengan membuat `RequestSpotInstanceRequest` objek. Objek permintaan memerlukan jumlah instance yang ingin Anda mulai dan harga penawaran. Selain itu, Anda perlu mengatur permintaan, yang mencakup jenis instans, ID AMI, dan grup keamanan yang ingin Anda gunakan. `LaunchSpecification` Setelah permintaan diisi, Anda memanggil `requestSpotInstances` metode pada `AmazonEC2Client` objek. Contoh berikut menunjukkan cara meminta Instance 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 lebih lanjut, silakan kunjungi Tutorial: Advanced Amazon EC2 Spot Request Management atau RequestSpotInstanceskelas di Referensi AWS SDK untuk Java API.

> **Note**
>
> Anda akan dikenakan biaya untuk Instans Spot apa pun yang benar-benar diluncurkan, jadi pastikan Anda membatalkan permintaan apa pun dan menghentikan instans apa pun 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 melakukan polling metode describeSpotInstancePermintaan untuk status ID permintaan Spot yang ingin kami pantau.

ID permintaan yang dibuat pada Langkah 2 disematkan dalam respons terhadap `requestSpotInstances` permintaan kami. Contoh kode berikut menunjukkan bagaimana untuk mengumpulkan permintaan IDs 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 status permintaan. Kemudian loop sampai permintaan tidak dalam keadaan “terbuka”. Perhatikan bahwa kami memantau status tidak “terbuka”, melainkan status, katakanlah, “aktif”, karena permintaan dapat langsung “ditutup” jika ada masalah dengan argumen permintaan Anda. Contoh kode berikut memberikan rincian tentang bagaimana 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 dikeluarkan ke layar. Dalam kedua kasus tersebut, kami dapat melanjutkan ke langkah berikutnya untuk membersihkan permintaan aktif apa pun dan menghentikan instance yang sedang berjalan.

Langkah 5: Membersihkan Permintaan dan Instans Spot Anda

Terakhir, kita perlu membersihkan permintaan dan instance kita. Penting untuk membatalkan permintaan yang belum dibayar dan menghentikan instance apa pun. Hanya membatalkan permintaan Anda tidak akan menghentikan instans Anda, yang berarti Anda akan terus membayarnya. Jika Anda menghentikan instans, permintaan Spot Anda mungkin dibatalkan, tetapi ada beberapa skenario seperti jika Anda menggunakan tawaran persisten di mana penghentian instans Anda tidak cukup untuk menghentikan permintaan Anda agar tidak dipenuhi kembali. Oleh karena itu, merupakan praktik terbaik untuk membatalkan tawaran aktif apa pun dan menghentikan instance yang sedang berjalan.

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 belum selesai, Anda memerlukan ID instans yang terkait dengan permintaan yang memulainya. Contoh kode berikut mengambil kode asli kami untuk memantau instance dan menambahkan `ArrayList` di mana kami menyimpan ID instance yang terkait dengan `describeInstance` respons.

```
// 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 instance IDs, disimpan di`ArrayList`, menghentikan setiap 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 Semuanya Bersama

Untuk menyatukan semua ini, kami menyediakan pendekatan yang lebih berorientasi objek yang menggabungkan langkah-langkah sebelumnya yang kami tunjukkan: menginisialisasi EC2 Klien, mengirimkan Permintaan Spot, menentukan kapan Permintaan Spot tidak lagi 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 `GettingStartedApp` kelas, yang memiliki metode utama di mana kami melakukan panggilan fungsi tingkat tinggi. Secara khusus, kami menginisialisasi `Requests` objek yang dijelaskan sebelumnya. Kami mengirimkan permintaan Instans Spot. Kemudian kami menunggu permintaan Spot mencapai status “Aktif”. Akhirnya, kami membersihkan permintaan dan instance.

Kode sumber lengkap untuk contoh ini dapat dilihat atau diunduh di GitHub.

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

Langkah Berikutnya

Lanjutkan dengan Tutorial: Advanced Amazon EC2 Spot Request Management.

Tutorial: Manajemen Permintaan Amazon EC2 Spot Tingkat Lanjut

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

Prasyarat

Untuk menggunakan tutorial ini, Anda harus AWS SDK untuk Java menginstal, serta telah memenuhi prasyarat instalasi dasarnya. Lihat Mengatur AWS SDK untuk Java untuk informasi lebih lanjut.

Menyiapkan kredensil Anda

Untuk mulai menggunakan contoh kode ini, Anda perlu mengatur AWS kredensil. Lihat Menyiapkan AWS Kredensial dan Wilayah untuk Pengembangan untuk petunjuk tentang cara melakukannya.

> **Note**
>
> Kami menyarankan Anda menggunakan kredensi IAM pengguna untuk memberikan nilai-nilai ini. Untuk informasi selengkapnya, lihat Mendaftar AWS dan Membuat IAM Pengguna.

Sekarang setelah Anda mengonfigurasi pengaturan Anda, Anda dapat mulai menggunakan kode dalam contoh.

Menyiapkan grup keamanan

Grup keamanan bertindak sebagai firewall yang mengontrol lalu lintas yang diizinkan masuk dan keluar dari sekelompok instance. Secara default, sebuah instance dimulai tanpa grup keamanan apa pun, yang berarti bahwa semua lalu lintas IP yang masuk, pada port TCP apa pun akan ditolak. Jadi, sebelum mengirimkan Permintaan Spot kami, kami akan membuat grup keamanan yang memungkinkan lalu lintas jaringan yang diperlukan. Untuk keperluan tutorial ini, kami 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 perlu menyertakan atau menjalankan contoh kode berikut yang mengatur grup keamanan secara terprogram.

Setelah kami membuat objek `AmazonEC2` klien, kami membuat `CreateSecurityGroupRequest` objek dengan nama, "GettingStarted" dan deskripsi untuk grup keamanan. Kemudian kita memanggil `ec2.createSecurityGroup` API untuk membuat grup.

Untuk mengaktifkan akses ke grup, kami membuat `ipPermission` objek dengan rentang alamat IP yang 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 ini sama dengan 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 dalam contoh `advanced.CreateSecurityGroupApp.java` kode. Catatan Anda hanya perlu menjalankan aplikasi ini sekali untuk membuat grup keamanan baru.

> **Note**
>
> Anda juga dapat membuat grup keamanan menggunakan file AWS Toolkit for Eclipse. Lihat Mengelola Grup Keamanan dari AWS Cost Explorer Panduan AWS Toolkit for Eclipse Pengguna untuk informasi selengkapnya.

Opsi pembuatan permintaan Instance Spot terperinci

Seperti yang kami jelaskan di Tutorial: Instans Amazon EC2 Spot, Anda perlu membuat permintaan Anda dengan jenis instans, Amazon Machine Image (AMI), dan harga tawaran maksimum.

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

(Kode berikut ini sama dengan 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 instance yang diminta dihentikan, permintaan akan ditutup. Permintaan persisten dipertimbangkan untuk dipenuhi setiap kali tidak ada Instance Spot yang berjalan untuk permintaan yang sama. Untuk menentukan jenis permintaan, Anda hanya perlu mengatur Type on the Spot request. 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 secara opsional menentukan lamanya waktu permintaan Anda akan tetap valid. Anda dapat menentukan waktu mulai dan berakhir untuk periode ini. Secara default, permintaan Spot akan dipertimbangkan untuk dipenuhi sejak dibuat hingga dipenuhi atau dibatalkan oleh Anda. Namun Anda dapat membatasi masa berlaku jika perlu. Contoh cara menentukan periode ini ditunjukkan dalam kode berikut.

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

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

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

// Set the valid start time to be two minutes from now.
Calendar cal = Calendar.getInstance();
```

```
cal.add(Calendar.MINUTE, 2);
requestRequest.setValidFrom(cal.getTime());

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

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

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

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

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

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

Mengelompokkan permintaan Instans Amazon EC2 Spot Anda

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

Jika Anda ingin memastikan Instans Spot Anda diluncurkan dan dihentikan bersama-sama, maka Anda memiliki opsi untuk memanfaatkan grup peluncuran. Grup peluncuran adalah label yang mengelompokkan serangkaian tawaran bersama. Semua instans 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 instance 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 sekumpulan instance bersama-sama di Availability Zone yang sama. Semua instans yang berbagi grup Availability Zone dan dipenuhi 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 Anda. Contoh kode berikut menunjukkan cara mengatur Availability Zone.

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

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

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

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

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

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

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

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

Terakhir, Anda dapat menentukan grup penempatan jika Anda menggunakan Instans Spot High Performance Computing (HPC), seperti instance komputasi cluster atau instance GPU cluster. Grup penempatan memberi Anda latensi yang lebih rendah dan konektivitas bandwidth tinggi antar 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 ditampilkan di bagian ini adalah opsional. Penting juga untuk disadari 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 mempertahankan partisi root setelah gangguan atau penghentian

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

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

Dalam kode yang ditambahkan, kita membuat `BlockDeviceMapping` objek dan mengatur yang terkait Amazon Elastic Block Store (Amazon EBS) ke Amazon EBS objek yang telah kita konfigurasi untuk `not` dihapus jika Instance 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 instance Anda saat startup, Anda juga dapat menggunakan pengaturan pemetaan perangkat blokir. Atau, jika Anda melampirkan partisi non-root, Anda dapat menentukan Amazon EBS volume Amazon yang ingin Anda lampirkan ke Instans Spot Anda setelah dilanjutkan. Anda melakukan ini hanya dengan menentukan ID snapshot di nama perangkat Anda `EbsBlockDevice` dan alternatif di objek Anda`BlockDeviceMapping`. Dengan memanfaatkan pemetaan perangkat blok, akan lebih mudah untuk mem-bootstrap instance Anda.

Menggunakan partisi root untuk memeriksa data penting Anda adalah cara yang bagus untuk mengelola potensi gangguan instance Anda. Untuk metode lebih lanjut tentang mengelola potensi interupsi, silakan kunjungi video Mengelola Gangguan.

Cara menandai permintaan dan instance spot Anda

Menambahkan tag ke Amazon EC2 sumber daya dapat menyederhanakan administrasi infrastruktur cloud Anda. Suatu bentuk metadata, tag dapat digunakan untuk membuat nama yang mudah digunakan, meningkatkan kemampuan pencarian, dan meningkatkan koordinasi antara beberapa pengguna. Anda juga dapat menggunakan tag untuk mengotomatiskan skrip dan bagian dari

proses Anda. Untuk membaca selengkapnya tentang penandaan Amazon EC2 sumber daya, buka Menggunakan Tag di Panduan Amazon EC2 Pengguna untuk Instans Linux.

Permintaan penandaan

Untuk menambahkan tag ke permintaan spot Anda, Anda perlu menandai mereka setelah diminta. Nilai pengembalian dari `requestSpotInstances()` memberi Anda RequestSpotInstancesResultobjek yang dapat Anda gunakan untuk mendapatkan permintaan spot IDs 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 memilikinya IDs, Anda dapat menandai permintaan dengan menambahkannya IDs ke CreateTagsRequestdan memanggil `createTags()` metode Amazon EC2 klien:

```
// 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());
}
```

Contoh penandaan

Demikian pula dengan permintaan spot itu sendiri, Anda hanya dapat menandai instance setelah dibuat, yang akan terjadi setelah permintaan spot dipenuhi (tidak lagi dalam keadaan terbuka).

Anda dapat memeriksa status permintaan Anda dengan memanggil `describeSpotInstanceRequests()` metode Amazon EC2 klien dengan DescribeSpotInstanceRequestsRequestobjek. DescribeSpotInstanceRequestsResultObjek yang dikembalikan berisi daftar SpotInstanceRequestobjek yang dapat Anda gunakan untuk menanyakan status permintaan spot Anda dan mendapatkan instance mereka IDs setelah mereka tidak lagi dalam keadaan terbuka.

Setelah permintaan spot tidak lagi terbuka, Anda dapat mengambil ID instance-nya dari `SpotInstanceRequest` objek dengan memanggil `getInstanceId()` metodenya.

```
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 instance 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 menghentikan instance

Membatalkan permintaan spot

Untuk membatalkan permintaan Instans Spot, panggil `cancelSpotInstanceRequests` Amazon EC2 klien dengan CancelSpotInstanceRequestsRequestobjek.

```
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 menghentikan Instans Spot apa pun yang berjalan dengan meneruskannya IDs ke metode Amazon EC2 klien. `terminateInstances()`

```
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());
}
```

Menyatukan semuanya

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 membuat instance kelas yang disebut `Requests` yang melakukan tindakan ini. Kami juga membuat `GettingStartedApp` kelas, yang memiliki metode utama di mana kami melakukan panggilan fungsi tingkat tinggi.

Kode sumber lengkap untuk contoh ini dapat dilihat atau diunduh di GitHub.

Selamat! Anda telah menyelesaikan tutorial Fitur Permintaan Lanjutan untuk mengembangkan perangkat lunak Instans Spot dengan fitur AWS SDK untuk Java.

Mengelola Amazon EC2 Instans

Membuat sebuah Instance

Buat Amazon EC2 instance baru dengan memanggil `runInstances` metode Amazon EC2 Client, menyediakannya dengan RunInstancesRequestberisi Amazon Machine Image (AMI) untuk digunakan dan jenis instance.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.InstanceType;
import com.amazonaws.services.ec2.model.RunInstancesRequest;
import com.amazonaws.services.ec2.model.RunInstancesResult;
import com.amazonaws.services.ec2.model.Tag;
```

Kode

```
RunInstancesRequest run_request = new RunInstancesRequest()
    .withImageId(ami_id)
    .withInstanceType(InstanceType.T1Micro)
    .withMaxCount(1)
    .withMinCount(1);

RunInstancesResult run_response = ec2.runInstances(run_request);

String reservation_id =
 run_response.getReservation().getInstances().get(0).getInstanceId();
```

Lihat contoh lengkapnya.

Memulai sebuah Instance

Untuk memulai sebuah Amazon EC2 instance, panggil `startInstances` metode Amazon EC2 Client, berikan dengan ID StartInstancesRequestyang berisi instance untuk memulai.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.StartInstancesRequest;
```

Kode

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

StartInstancesRequest request = new StartInstancesRequest()
    .withInstanceIds(instance_id);

ec2.startInstances(request);
```

Lihat contoh lengkapnya.

Menghentikan sebuah Instance

Untuk menghentikan Amazon EC2 instance, panggil `stopInstances` metode EC2 Klien Amazon, berikan ID yang StopInstancesRequestberisi instance untuk berhenti.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.StopInstancesRequest;
```

Kode

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

StopInstancesRequest request = new StopInstancesRequest()
    .withInstanceIds(instance_id);
```

```
ec2.stopInstances(request);
```

Lihat contoh lengkapnya.

Mem-boot Ulang Instans

Untuk me-reboot sebuah Amazon EC2 instance, panggil `rebootInstances` metode Amazon EC2 Client, berikan ID RebootInstancesRequestyang berisi instance untuk reboot.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.RebootInstancesRequest;
import com.amazonaws.services.ec2.model.RebootInstancesResult;
```

Kode

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

RebootInstancesRequest request = new RebootInstancesRequest()
    .withInstanceIds(instance_id);

RebootInstancesResult response = ec2.rebootInstances(request);
```

Lihat contoh lengkapnya.

Menjelaskan Instans

Untuk membuat daftar instance Anda, buat DescribeInstancesRequestdan panggil `describeInstances` metode EC2 Klien Amazon. Ini akan mengembalikan DescribeInstancesResultobjek yang dapat Anda gunakan untuk daftar Amazon EC2 instance untuk akun dan wilayah Anda.

Instans dikelompokkan berdasarkan reservasi. Setiap reservasi sesuai dengan panggilan `startInstances` yang meluncurkan instance. Untuk membuat daftar instans Anda, Anda harus terlebih dahulu memanggil `DescribeInstancesResult` kelas `getReservations' method, and then call `getInstances` pada setiap objek Reservasi yang dikembalikan.

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 paged; Anda bisa mendapatkan hasil lebih lanjut dengan meneruskan nilai yang dikembalikan dari `getNextToken` metode objek hasil ke metode objek permintaan asli Anda, kemudian menggunakan objek permintaan yang sama dalam panggilan berikutnya. `setNextToken` `describeInstances`

Lihat contoh lengkapnya.

Memantau sebuah Instance

Anda dapat memantau berbagai aspek Amazon EC2 instance Anda, seperti CPU dan pemanfaatan jaringan, memori yang tersedia, dan ruang disk yang tersisa. Untuk mempelajari lebih lanjut tentang pemantauan instans, lihat Pemantauan Amazon EC2 di Panduan Amazon EC2 Pengguna untuk Instans Linux.

Untuk mulai memantau instance, Anda harus membuat MonitorInstancesRequestdengan ID instance untuk dipantau, dan meneruskannya ke `monitorInstances` metode EC2 Klien Amazon.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.MonitorInstancesRequest;
```

Kode

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

MonitorInstancesRequest request = new MonitorInstancesRequest()
        .withInstanceIds(instance_id);

ec2.monitorInstances(request);
```

Lihat contoh lengkapnya.

Menghentikan Pemantauan Instance

Untuk menghentikan pemantauan instance, buat UnmonitorInstancesRequestdengan ID instance untuk menghentikan pemantauan, dan teruskan ke `unmonitorInstances` metode EC2 Klien Amazon.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.UnmonitorInstancesRequest;
```

Kode

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

UnmonitorInstancesRequest request = new UnmonitorInstancesRequest()
    .withInstanceIds(instance_id);

ec2.unmonitorInstances(request);
```

Lihat contoh lengkapnya.

Informasi Selengkapnya

- RunInstancesdi Referensi Amazon EC2 API
- DescribeInstancesdi Referensi Amazon EC2 API
- StartInstancesdi Referensi Amazon EC2 API
- StopInstancesdi Referensi Amazon EC2 API
- RebootInstancesdi Referensi Amazon EC2 API
- MonitorInstancesdi Referensi Amazon EC2 API
- UnmonitorInstancesdi Referensi Amazon EC2 API

Menggunakan Alamat IP Elastis di Amazon EC2

EC2-Classic pensiun

> **Warning**
>
> Kami pensiun EC2 -Classic pada 15 Agustus 2022. Kami menyarankan Anda bermigrasi dari EC2 -Classic ke VPC. Untuk informasi lebih lanjut, lihat posting blog EC2-Classic-Classic Networking is Retiring - Inilah Cara Mempersiapkan.

Mengalokasikan Alamat IP Elastis

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

Untuk mengalokasikan alamat IP Elastis, panggil `allocateAddress` metode EC2 Klien Amazon dengan AllocateAddressRequestobjek yang berisi tipe jaringan (klasik EC2 atau VPC).

Yang dikembalikan AllocateAddressResultberisi ID alokasi yang dapat Anda gunakan untuk mengaitkan alamat dengan instance, dengan meneruskan ID alokasi dan ID instance AssociateAddressRequestke metode EC2 Klien Amazon`associateAddress`.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.AllocateAddressRequest;
import com.amazonaws.services.ec2.model.AllocateAddressResult;
import com.amazonaws.services.ec2.model.AssociateAddressRequest;
import com.amazonaws.services.ec2.model.AssociateAddressResult;
import com.amazonaws.services.ec2.model.DomainType;
```

Kode

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

AllocateAddressRequest allocate_request = new AllocateAddressRequest()
    .withDomain(DomainType.Vpc);

AllocateAddressResult allocate_response =
    ec2.allocateAddress(allocate_request);

String allocation_id = allocate_response.getAllocationId();

AssociateAddressRequest associate_request =
    new AssociateAddressRequest()
        .withInstanceId(instance_id)
        .withAllocationId(allocation_id);

AssociateAddressResult associate_response =
    ec2.associateAddress(associate_request);
```

Lihat contoh lengkapnya.

Menjelaskan Alamat IP Elastis

Untuk mencantumkan alamat IP Elastis yang ditetapkan ke akun Anda, hubungi `describeAddresses` metode EC2 Klien Amazon. Ini mengembalikan DescribeAddressesResultyang dapat Anda gunakan untuk mendapatkan daftar objek Alamat yang mewakili alamat IP Elastis di akun Anda.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.Address;
import com.amazonaws.services.ec2.model.DescribeAddressesResult;
```

Kode

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

DescribeAddressesResult response = ec2.describeAddresses();

for(Address address : response.getAddresses()) {
    System.out.printf(
            "Found address with public IP %s, " +
            "domain %s, " +
            "allocation id %s " +
            "and NIC id %s",
            address.getPublicIp(),
            address.getDomain(),
            address.getAllocationId(),
            address.getNetworkInterfaceId());
}
```

Lihat contoh lengkapnya.

Melepaskan Alamat IP Elastis

Untuk merilis alamat IP Elastis, panggil `releaseAddress` metode EC2 Klien Amazon, berikan ID alokasi yang ReleaseAddressRequestberisi 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, itu dilepaskan ke kumpulan alamat AWS IP dan mungkin tidak tersedia untuk Anda sesudahnya. Pastikan untuk memperbarui catatan DNS Anda dan server atau perangkat apa pun yang berkomunikasi dengan alamat tersebut. Jika Anda mencoba melepaskan alamat IP Elastis yang sudah dirilis, Anda akan mendapatkan AuthFailurekesalahan jika alamat tersebut sudah dialokasikan ke alamat lain Akun AWS.

Jika Anda menggunakan EC2-Classic atau VPC default, maka melepaskan alamat IP Elastis secara otomatis memisahkannya dari instance apa pun yang terkait dengannya. Untuk memisahkan alamat IP Elastis tanpa melepaskannya, gunakan metode EC2 Klien Amazon. `disassociateAddress`

Jika Anda menggunakan VPC non-default, Anda harus `disassociateAddress` menggunakan untuk memisahkan alamat IP Elastic sebelum Anda mencoba melepaskannya. Jika tidak, Amazon EC2 mengembalikan kesalahan (Tidak validIPAddress. InUse).

Lihat contoh lengkapnya.

Informasi Selengkapnya

- Alamat IP Elastis dalam Panduan Amazon EC2 Pengguna untuk Instans Linux
- AllocateAddressdi Referensi Amazon EC2 API
- DescribeAddressesdi Referensi Amazon EC2 API
- ReleaseAddressdi Referensi Amazon EC2 API

Gunakan wilayah dan zona ketersediaan

Jelaskan wilayah

Untuk mencantumkan Wilayah yang tersedia untuk akun Anda, hubungi `describeRegions` metode EC2 Klien Amazon. Ini mengembalikan a DescribeRegionsResult. Panggil `getRegions` metode objek yang dikembalikan untuk mendapatkan daftar objek Region yang mewakili setiap Region.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeRegionsResult;
import com.amazonaws.services.ec2.model.Region;
import com.amazonaws.services.ec2.model.AvailabilityZone;
import com.amazonaws.services.ec2.model.DescribeAvailabilityZonesResult;
```

Kode

```
DescribeRegionsResult regions_response = ec2.describeRegions();

for(Region region : regions_response.getRegions()) {
    System.out.printf(
        "Found region %s " +
        "with endpoint %s",
        region.getRegionName(),
        region.getEndpoint());
}
```

Lihat contoh lengkapnya.

Jelaskan zona ketersediaan

Untuk mencantumkan setiap Availability Zone yang tersedia untuk akun Anda, hubungi `describeAvailabilityZones` metode Amazon EC2 Client. Ini mengembalikan a DescribeAvailabilityZonesResult. Panggil `getAvailabilityZones` metodenya untuk mendapatkan daftar AvailabilityZoneobjek yang mewakili setiap Availability Zone.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeRegionsResult;
import com.amazonaws.services.ec2.model.Region;
import com.amazonaws.services.ec2.model.AvailabilityZone;
import com.amazonaws.services.ec2.model.DescribeAvailabilityZonesResult;
```

Kode

```
DescribeAvailabilityZonesResult zones_response =
    ec2.describeAvailabilityZones();
```

```
for(AvailabilityZone zone : zones_response.getAvailabilityZones()) {
    System.out.printf(
        "Found availability zone %s " +
        "with status %s " +
        "in region %s",
        zone.getZoneName(),
        zone.getState(),
        zone.getRegionName());
}
```

Lihat contoh lengkapnya.

Jelaskan akun

Untuk mendeskripsikan akun Anda, hubungi `describeAccountAttributes` metode EC2 Klien Amazon. Metode ini mengembalikan DescribeAccountAttributesResultobjek. Memanggil `getAccountAttributes` metode objek ini untuk mendapatkan daftar AccountAttributeobjek. Anda dapat mengulangi melalui daftar untuk mengambil objek AccountAttribute.

Anda bisa mendapatkan nilai atribut akun Anda dengan menjalankan `getAttributeValues` metode AccountAttributeobjek. Metode ini mengembalikan daftar AccountAttributeValueobjek. Anda dapat mengulangi 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();
}
```

Lihat contoh lengkapnya di GitHub.

Informasi lain

- Wilayah dan Availability Zone di Panduan Amazon EC2 Pengguna untuk Instans Linux
- DescribeRegionsdi Referensi Amazon EC2 API
- DescribeAvailabilityZonesdi Referensi Amazon EC2 API

Bekerja dengan Pasangan Amazon EC2 Kunci

Membuat Pasangan Kunci

Untuk membuat key pair, panggil `createKeyPair` metode Amazon EC2 Client dengan CreateKeyPairRequestyang 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 contoh lengkapnya.

Menggambarkan Pasangan Kunci

Untuk membuat daftar pasangan kunci Anda atau untuk mendapatkan informasi tentang mereka, hubungi `describeKeyPairs` metode EC2 Klien Amazon. Ia mengembalikan DescribeKeyPairsResultyang dapat Anda gunakan untuk mengakses daftar pasangan kunci dengan memanggil `getKeyPairs` metodenya, yang mengembalikan daftar KeyPairInfoobjek.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeKeyPairsResult;
import com.amazonaws.services.ec2.model.KeyPairInfo;
```

Kode

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

DescribeKeyPairsResult response = ec2.describeKeyPairs();

for(KeyPairInfo key_pair : response.getKeyPairs()) {
    System.out.printf(
        "Found key pair with name %s " +
        "and fingerprint %s",
        key_pair.getKeyName(),
        key_pair.getKeyFingerprint());
}
```

Lihat contoh lengkapnya.

Menghapus Pasangan Kunci

Untuk menghapus key pair, panggil `deleteKeyPair` metode Amazon EC2 Client, berikan DeleteKeyPairRequestyang berisi nama key pair yang akan dihapus.

Impor

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DeleteKeyPairRequest;
import com.amazonaws.services.ec2.model.DeleteKeyPairResult;
```

Kode

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

DeleteKeyPairRequest request = new DeleteKeyPairRequest()
    .withKeyName(key_name);

DeleteKeyPairResult response = ec2.deleteKeyPair(request);
```

Lihat contoh lengkapnya.

Informasi Selengkapnya

- Amazon EC2 Pasangan Kunci dalam Panduan Amazon EC2 Pengguna untuk Instans Linux
- CreateKeyPairdi Referensi Amazon EC2 API
- DescribeKeyPairsdi Referensi Amazon EC2 API
- DeleteKeyPairdi Referensi Amazon EC2 API

Bekerja dengan kelompok keamanan di Amazon EC2

Membuat Grup Keamanan

Untuk membuat grup keamanan, panggil `createSecurityGroup` metode EC2 Klien Amazon 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.

Mengkonfigurasi Grup Keamanan

Grup keamanan dapat mengontrol lalu lintas masuk (masuk) dan keluar (keluar) ke instans Anda. Amazon EC2

Untuk menambahkan aturan ingress ke grup keamanan Anda, gunakan `authorizeSecurityGroupIngress` metode EC2 Klien Amazon, dengan memberikan nama grup keamanan dan aturan akses (IpPermission) yang ingin Anda tetapkan padanya dalam objek AuthorizeSecurityGroupIngressRequest. Contoh berikut menunjukkan cara menambahkan izin IP ke grup keamanan.

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 keluar ke grup keamanan, berikan data serupa dalam metode AuthorizeSecurityGroupEgressRequestke EC2 Klien AmazonauthorizeSecurityGroupEgress.

Lihat contoh lengkapnya.

Menggambarkan Grup Keamanan

Untuk menjelaskan grup keamanan Anda atau mendapatkan informasi tentang mereka, hubungi `describeSecurityGroups` metode EC2 Klien Amazon. Ia mengembalikan DescribeSecurityGroupsResultyang dapat Anda gunakan untuk mengakses daftar grup 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 Grup Keamanan

Untuk menghapus grup keamanan, panggil `deleteSecurityGroup` metode EC2 Klien Amazon, berikan DeleteSecurityGroupRequestyang berisi ID grup keamanan yang akan 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 EC2 Grup Keamanan dalam Panduan Amazon EC2 Pengguna untuk Instans Linux
- Mengotorisasi Lalu Lintas Masuk untuk Instans Linux Anda di Panduan Amazon EC2 Pengguna untuk Instans Linux

- CreateSecurityGroupdi Referensi Amazon EC2 API
- DescribeSecurityGroupsdi Referensi Amazon EC2 API
- DeleteSecurityGroupdi Referensi Amazon EC2 API
- AuthorizeSecurityGroupIngressdi Referensi Amazon EC2 API

Contoh IAM Menggunakan AWS SDK untuk Java

Bagian ini memberikan contoh pemrograman IAM dengan menggunakan. AWS SDK untuk Java

AWS Identity and Access Management (IAM) memungkinkan Anda untuk mengontrol akses ke AWS layanan dan sumber daya untuk pengguna Anda dengan aman. Menggunakan IAM, Anda dapat membuat dan mengelola AWS pengguna dan grup, dan menggunakan izin untuk mengizinkan dan menolak akses mereka ke AWS sumber daya. Untuk panduan lengkap IAM, kunjungi Panduan IAM Pengguna.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. Kode contoh lengkap tersedia di GitHub. Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

- Mengelola Kunci Akses IAM
- Mengelola Pengguna IAM
- Menggunakan Alias Akun IAM
- Bekerja dengan Kebijakan IAM
- Bekerja dengan Sertifikat Server IAM

Mengelola Kunci Akses IAM

Membuat Kunci Akses

Untuk membuat kunci akses IAM, panggil AmazonIdentityManagementClient `createAccessKey` metode dengan CreateAccessKeyRequestobjek.

`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 metode `withUserName` 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 contoh lengkapnya di GitHub.

Kunci Akses Daftar

Untuk membuat daftar kunci akses untuk pengguna tertentu, buat ListAccessKeysRequestobjek yang berisi nama pengguna untuk mencantumkan kunci, dan meneruskannya ke AmazonIdentityManagementClient `listAccessKeys` metode.

> **Note**
>
> Jika Anda tidak memberikan nama pengguna ke`listAccessKeys`, itu akan mencoba untuk daftar kunci akses yang terkait dengan Akun AWS yang menandatangani permintaan.

Impor

```
import 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` paged (dengan maksimum default 100 record per panggilan). Anda dapat memanggil `getIsTruncated` ListAccessKeysResultobjek yang dikembalikan untuk melihat apakah kueri mengembalikan hasil yang lebih sedikit kemudian tersedia. Jika demikian, maka panggil `ListAccessKeysRequest` dan `setMarker` teruskan kembali ke pemanggilan berikutnya. `listAccessKeys`

Lihat contoh lengkapnya di GitHub.

Mengambil Waktu Terakhir Digunakan Kunci Akses

Untuk mendapatkan waktu kunci akses terakhir digunakan, panggil `getAccessKeyLastUsed` metode AmazonIdentityManagementClient ini dengan ID kunci akses (yang dapat diteruskan

menggunakan GetAccessKeyLastUsedRequestobjek, atau langsung ke overload yang mengambil ID kunci akses secara langsung.

Anda kemudian dapat menggunakan GetAccessKeyLastUsedResultobjek yang dikembalikan untuk mengambil waktu terakhir kunci yang digunakan.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetAccessKeyLastUsedRequest;
import com.amazonaws.services.identitymanagement.model.GetAccessKeyLastUsedResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

GetAccessKeyLastUsedRequest request = new GetAccessKeyLastUsedRequest()
    .withAccessKeyId(access_id);

GetAccessKeyLastUsedResult response = iam.getAccessKeyLastUsed(request);

System.out.println("Access key was last used at: " +
        response.getAccessKeyLastUsed().getLastUsedDate());
```

Lihat contoh lengkapnya di GitHub.

Mengaktifkan atau Menonaktifkan Kunci Akses

Anda dapat mengaktifkan atau menonaktifkan kunci akses dengan membuat UpdateAccessKeyRequestobjek, memberikan ID kunci akses, opsional nama pengguna, dan Status yang diinginkan, lalu meneruskan objek permintaan ke metode AmazonIdentityManagementClient iniupdateAccessKey.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.UpdateAccessKeyRequest;
import com.amazonaws.services.identitymanagement.model.UpdateAccessKeyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

UpdateAccessKeyRequest request = new UpdateAccessKeyRequest()
    .withAccessKeyId(access_id)
    .withUserName(username)
    .withStatus(status);

UpdateAccessKeyResult response = iam.updateAccessKey(request);
```

Lihat contoh lengkapnya di GitHub.

Menghapus Kunci Akses

Untuk menghapus kunci akses secara permanen, panggil `deleteKey` metode ini, berikan dengan ID dan nama pengguna kunci akses yang DeleteAccessKeyRequestberisi. AmazonIdentityManagementClient

> **Note**
>
> Setelah dihapus, kunci tidak dapat lagi diambil atau digunakan. Untuk menonaktifkan sementara kunci sehingga dapat diaktifkan lagi nanti, gunakan updateAccessKeymetode sebagai gantinya.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteAccessKeyRequest;
import com.amazonaws.services.identitymanagement.model.DeleteAccessKeyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DeleteAccessKeyRequest request = new DeleteAccessKeyRequest()
```

```
    .withAccessKeyId(access_key)
    .withUserName(username);

DeleteAccessKeyResult response = iam.deleteAccessKey(request);
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- CreateAccessKeydi Referensi API IAM
- ListAccessKeysdi Referensi API IAM
- GetAccessKeyLastUseddi Referensi API IAM
- UpdateAccessKeydi Referensi API IAM
- DeleteAccessKeydi Referensi API IAM

Mengelola Pengguna IAM

Membuat Pengguna

Buat pengguna IAM baru dengan memberikan nama pengguna ke `createUser` metode AmazonIdentityManagementClient 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 lengkapnya di GitHub.

Daftar Pengguna

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

Daftar pengguna yang dikembalikan `listUsers` oleh halaman. Anda dapat memeriksa untuk melihat ada lebih banyak hasil untuk diambil dengan memanggil `getIsTruncated` metode objek respons. Jika kembali`true`, kemudian memanggil `setMarker()` metode permintaan objek, melewati nilai kembali dari `getMarker()` metode objek respon.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListUsersRequest;
import com.amazonaws.services.identitymanagement.model.ListUsersResult;
import com.amazonaws.services.identitymanagement.model.User;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

boolean done = false;
ListUsersRequest request = new ListUsersRequest();

while(!done) {
    ListUsersResult response = iam.listUsers(request);

    for(User user : response.getUsers()) {
        System.out.format("Retrieved user %s", user.getUserName());
    }

    request.setMarker(response.getMarker());

    if(!response.getIsTruncated()) {
```

```
        done = true;
    }
}
```

Lihat contoh lengkapnya di GitHub.

Memperbarui Pengguna

Untuk memperbarui pengguna, panggil `updateUser` metode AmazonIdentityManagementClient 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 lengkapnya di GitHub.

Menghapus Pengguna

Untuk menghapus pengguna, panggil AmazonIdentityManagementClient `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 lengkapnya di GitHub.

Informasi Selengkapnya

- Pengguna IAM dalam IAM Panduan Pengguna
- Mengelola Pengguna IAM di IAM Panduan Pengguna
- CreateUserdi Referensi API IAM
- ListUsersdi Referensi API IAM
- UpdateUserdi Referensi API IAM
- DeleteUserdi Referensi API IAM

Menggunakan Alias Akun IAM

Jika Anda ingin URL untuk halaman login berisi nama perusahaan atau pengenal ramah lainnya, bukan Akun AWS ID Anda, Anda dapat membuat alias untuk Anda. Akun AWS

> **Note**
>
> AWS mendukung persis satu alias akun per akun.

Membuat Akun Alias

Untuk membuat alias akun, panggil `createAccountAlias` metode AmazonIdentityManagementClient ini dengan CreateAccountAliasRequestobjek yang berisi nama alias.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateAccountAliasRequest;
import com.amazonaws.services.identitymanagement.model.CreateAccountAliasResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

CreateAccountAliasRequest request = new CreateAccountAliasRequest()
    .withAccountAlias(alias);

CreateAccountAliasResult response = iam.createAccountAlias(request);
```

Lihat contoh lengkapnya di GitHub.

Daftar Alias Akun

Untuk membuat daftar alias akun Anda, jika ada, hubungi `listAccountAliases` metode AmazonIdentityManagementClient ini.

> **Note**
>
> Yang dikembalikan ListAccountAliasesResultmendukung yang sama `getIsTruncated` dan `getMarker` metode seperti metode AWS SDK untuk Java daftar lainnya, tetapi hanya Akun AWS dapat memiliki satu alias akun.

impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
```

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListAccountAliasesResult;
```

kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

ListAccountAliasesResult response = iam.listAccountAliases();

for (String alias : response.getAccountAliases()) {
    System.out.printf("Retrieved account alias %s", alias);
}
```

lihat contoh lengkapnya di GitHub.

Menghapus alias akun

Untuk menghapus alias akun Anda, hubungi `deleteAccountAlias` metode AmazonIdentityManagementClient ini. Saat menghapus alias akun, Anda harus memberikan namanya menggunakan objek. DeleteAccountAliasRequest

impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteAccountAliasRequest;
import com.amazonaws.services.identitymanagement.model.DeleteAccountAliasResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DeleteAccountAliasRequest request = new DeleteAccountAliasRequest()
    .withAccountAlias(alias);

DeleteAccountAliasResult response = iam.deleteAccountAlias(request);
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- ID AWS Akun Anda dan Aliasnya di IAM Panduan Pengguna
- CreateAccountAliasdi Referensi API IAM
- ListAccountAliasesdi Referensi API IAM
- DeleteAccountAliasdi Referensi API IAM

Bekerja dengan Kebijakan IAM

Membuat Kebijakan

Untuk membuat kebijakan baru, berikan nama kebijakan dan dokumen kebijakan berformat JSON dalam metode CreatePolicyRequestto the's. AmazonIdentityManagementClient `createPolicy`

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreatePolicyRequest;
import com.amazonaws.services.identitymanagement.model.CreatePolicyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

CreatePolicyRequest request = new CreatePolicyRequest()
    .withPolicyName(policy_name)
    .withPolicyDocument(POLICY_DOCUMENT);

CreatePolicyResult response = iam.createPolicy(request);
```

Dokumen kebijakan IAM adalah string JSON dengan sintaks yang terdokumentasi dengan baik.
Berikut adalah contoh yang menyediakan akses untuk membuat permintaan tertentu DynamoDB.

```
public static final String POLICY_DOCUMENT =
    "{" +
    "  \"Version\": \"2012-10-17\",       " +
```

```
    "  \"Statement\": [" +
    "    {" +
    "        \"Effect\": \"Allow\"," +
    "        \"Action\": \"logs:CreateLogGroup\"," +
    "        \"Resource\": \"%s\"" +
    "    }," +
    "    {" +
    "        \"Effect\": \"Allow\"," +
    "        \"Action\": [" +
    "            \"dynamodb:DeleteItem\"," +
    "            \"dynamodb:GetItem\"," +
    "            \"dynamodb:PutItem\"," +
    "            \"dynamodb:Scan\"," +
    "            \"dynamodb:UpdateItem\"" +
    "       ]," +
    "       \"Resource\": \"RESOURCE_ARN\"" +
    "    }" +
    "   ]" +
    "}";
```

Lihat contoh lengkapnya di GitHub.

Mendapatkan Kebijakan

Untuk mengambil kebijakan yang ada, panggil `getPolicy` metode, yang menyediakan ARN kebijakan dalam GetPolicyRequestobjek. AmazonIdentityManagementClient

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetPolicyRequest;
import com.amazonaws.services.identitymanagement.model.GetPolicyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

GetPolicyRequest request = new GetPolicyRequest()
    .withPolicyArn(policy_arn);
```

```
GetPolicyResult response = iam.getPolicy(request);
```

Lihat contoh lengkapnya di GitHub.

Melampirkan Kebijakan Peran

Anda dapat melampirkan kebijakan ke IAMhttp: //docs.aws.amazon. com/IAM/latest/UserGuide/id_roles.html [role] dengan memanggil `attachRolePolicy` metode AmazonIdentityManagementClient's, menyediakannya dengan nama peran dan kebijakan ARN dalam file. AttachRolePolicyRequest

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.AttachRolePolicyRequest;
import com.amazonaws.services.identitymanagement.model.AttachedPolicy;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

AttachRolePolicyRequest attach_request =
    new AttachRolePolicyRequest()
        .withRoleName(role_name)
        .withPolicyArn(POLICY_ARN);

iam.attachRolePolicy(attach_request);
```

Lihat contoh lengkapnya di GitHub.

Daftar Kebijakan Peran Terlampir

Buat daftar kebijakan terlampir pada peran dengan memanggil `listAttachedRolePolicies` metode ini. AmazonIdentityManagementClient Dibutuhkan ListAttachedRolePoliciesRequestobjek yang berisi nama peran untuk mencantumkan kebijakan.

Panggil `getAttachedPolicies` ListAttachedRolePoliciesResultobjek yang dikembalikan untuk mendapatkan daftar kebijakan terlampir. Hasil mungkin terpotong; jika `getIsTruncated`

metode `ListAttachedRolePoliciesResult` objek kembali`true`, panggil `setMarker` metode `ListAttachedRolePoliciesRequest` objek dan gunakan untuk memanggil `listAttachedRolePolicies` lagi untuk mendapatkan kumpulan hasil berikutnya.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListAttachedRolePoliciesRequest;
import com.amazonaws.services.identitymanagement.model.ListAttachedRolePoliciesResult;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

ListAttachedRolePoliciesRequest request =
    new ListAttachedRolePoliciesRequest()
        .withRoleName(role_name);

List<AttachedPolicy> matching_policies = new ArrayList<>();

boolean done = false;

while(!done) {
    ListAttachedRolePoliciesResult response =
        iam.listAttachedRolePolicies(request);

    matching_policies.addAll(
            response.getAttachedPolicies()
                    .stream()
                    .filter(p -> p.getPolicyName().equals(role_name))
                    .collect(Collectors.toList()));

    if(!response.getIsTruncated()) {
        done = true;
    }
    request.setMarker(response.getMarker());
}
```

Lihat contoh lengkapnya di GitHub.

Melepaskan Kebijakan Peran

Untuk melepaskan kebijakan dari peran, panggil `detachRolePolicy` metode AmazonIdentityManagementClient ini, berikan nama peran dan kebijakan ARN dalam file. DetachRolePolicyRequest

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DetachRolePolicyRequest;
import com.amazonaws.services.identitymanagement.model.DetachRolePolicyResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DetachRolePolicyRequest request = new DetachRolePolicyRequest()
    .withRoleName(role_name)
    .withPolicyArn(policy_arn);

DetachRolePolicyResult response = iam.detachRolePolicy(request);
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- Ikhtisar Kebijakan IAM dalam Panduan IAM Pengguna.
- AWS Referensi Kebijakan IAM dalam Panduan IAM Pengguna.
- CreatePolicydi Referensi API IAM
- GetPolicydi Referensi API IAM
- AttachRolePolicydi Referensi API IAM
- ListAttachedRolePoliciesdi Referensi API IAM
- DetachRolePolicydi Referensi API IAM

Bekerja dengan Sertifikat Server IAM

Untuk mengaktifkan koneksi HTTPS ke situs web atau aplikasi Anda AWS, Anda memerlukan sertifikat server SSL/TLS. Anda dapat menggunakan sertifikat server yang disediakan oleh AWS Certificate Manager atau sertifikat yang Anda peroleh dari penyedia eksternal.

Kami menyarankan Anda menggunakan ACM untuk menyediakan, mengelola, dan menyebarkan sertifikat server Anda. Dengan ACM Anda dapat meminta sertifikat, menyebarkannya ke AWS sumber daya Anda, dan membiarkan ACM menangani perpanjangan sertifikat untuk Anda. Sertifikat yang disediakan oleh ACM gratis. Untuk informasi selengkapnya tentang ACM, lihat Panduan Pengguna ACM.

Mendapatkan Sertifikat Server

Anda dapat mengambil sertifikat server dengan memanggil `getServerCertificate` metode, meneruskannya GetServerCertificateRequestdengan nama sertifikat. AmazonIdentityManagementClient

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.GetServerCertificateRequest;
import com.amazonaws.services.identitymanagement.model.GetServerCertificateResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

GetServerCertificateRequest request = new GetServerCertificateRequest()
            .withServerCertificateName(cert_name);

GetServerCertificateResult response = iam.getServerCertificate(request);
```

Lihat contoh lengkapnya di GitHub.

Daftar Sertifikat Server

Untuk mencantumkan sertifikat server Anda, panggil `listServerCertificates` metode ini dengan file ListServerCertificatesRequest. AmazonIdentityManagementClient Ini mengembalikan a ListServerCertificatesResult.

Panggil `getServerCertificateMetadataList` metode `ListServerCertificateResult` objek yang dikembalikan untuk mendapatkan daftar ServerCertificateMetadataobjek yang dapat Anda gunakan untuk mendapatkan informasi tentang setiap sertifikat.

Hasil mungkin terpotong; jika `getIsTruncated` metode `ListServerCertificateResult` objek kembali`true`, panggil `setMarker` metode `ListServerCertificatesRequest` objek dan gunakan untuk memanggil `listServerCertificates` lagi untuk mendapatkan kumpulan hasil berikutnya.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.ListServerCertificatesRequest;
import com.amazonaws.services.identitymanagement.model.ListServerCertificatesResult;
import com.amazonaws.services.identitymanagement.model.ServerCertificateMetadata;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

boolean done = false;
ListServerCertificatesRequest request =
        new ListServerCertificatesRequest();

while(!done) {

    ListServerCertificatesResult response =
        iam.listServerCertificates(request);

    for(ServerCertificateMetadata metadata :
            response.getServerCertificateMetadataList()) {
        System.out.printf("Retrieved server certificate %s",
                metadata.getServerCertificateName());
```

```
    }

    request.setMarker(response.getMarker());

    if(!response.getIsTruncated()) {
        done = true;
    }
}
```

Lihat contoh lengkapnya di GitHub.

Memperbarui Sertifikat Server

Anda dapat memperbarui nama atau jalur sertifikat server dengan memanggil `updateServerCertificate` metode ini. AmazonIdentityManagementClient Dibutuhkan UpdateServerCertificateRequestobjek yang ditetapkan dengan nama sertifikat server saat ini dan baik nama baru atau jalur baru untuk digunakan.

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.UpdateServerCertificateRequest;
import com.amazonaws.services.identitymanagement.model.UpdateServerCertificateResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

UpdateServerCertificateRequest request =
    new UpdateServerCertificateRequest()
        .withServerCertificateName(cur_name)
        .withNewServerCertificateName(new_name);

UpdateServerCertificateResult response =
    iam.updateServerCertificate(request);
```

Lihat contoh lengkapnya di GitHub.

Menghapus Sertifikat Server

Untuk menghapus sertifikat server, panggil `deleteServerCertificate` metode dengan DeleteServerCertificateRequestberisi nama sertifikat. AmazonIdentityManagementClient

Impor

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteServerCertificateRequest;
import com.amazonaws.services.identitymanagement.model.DeleteServerCertificateResult;
```

Kode

```
final AmazonIdentityManagement iam =
    AmazonIdentityManagementClientBuilder.defaultClient();

DeleteServerCertificateRequest request =
    new DeleteServerCertificateRequest()
        .withServerCertificateName(cert_name);

DeleteServerCertificateResult response =
    iam.deleteServerCertificate(request);
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- Bekerja dengan Sertifikat Server di Panduan IAM Pengguna
- GetServerCertificatedi Referensi API IAM
- ListServerCertificatesdi Referensi API IAM
- UpdateServerCertificatedi Referensi API IAM
- DeleteServerCertificatedi Referensi API IAM
- Panduan Pengguna ACM

Lambda Contoh Menggunakan AWS SDK untuk Java

Bagian ini memberikan contoh pemrograman Lambda menggunakan AWS SDK untuk Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. Kode contoh lengkap tersedia di GitHub. Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

Memanggil, Membuat Daftar, dan Menghapus Fungsi Lambda

Bagian ini memberikan contoh pemrograman dengan klien Lambda layanan dengan menggunakan AWS SDK untuk Java. Untuk mempelajari cara membuat Lambda fungsi, lihat Cara Membuat AWS Lambda fungsi.

Topik

Memanggil fungsi

Anda dapat memanggil Lambda fungsi dengan membuat AWSLambdaobjek dan menjalankan metodenya`invoke`. Buat InvokeRequestobjek untuk menentukan informasi tambahan seperti nama fungsi dan payload untuk diteruskan ke Lambda fungsi. Nama fungsi muncul sebagai arn:aws:lambda: us-east- 1:555556330391:function:. HelloFunction Anda dapat mengambil nilai dengan melihat fungsi di. Konsol Manajemen AWS

Untuk meneruskan data payload ke fungsi, panggil `withPayload` metode InvokeRequestobjek dan tentukan 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 bagaimana untuk memanggil fungsi. Lambda

```
        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 lengkapnya di Github.

Daftar fungsi

Bangun AWSLambdaobjek dan panggil `listFunctions` metodenya. Metode ini mengembalikan ListFunctionsResultobjek. Anda dapat memanggil `getFunctions` metode objek ini untuk mengembalikan daftar FunctionConfigurationobjek. Anda dapat mengulangi melalui daftar untuk mengambil informasi tentang fungsi. Misalnya, contoh kode Java berikut menunjukkan cara mendapatkan setiap nama fungsi.

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 menunjukkan bagaimana untuk mengambil daftar nama Lambda 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 lengkapnya di Github.

Hapus fungsi

Bangun AWSLambdaobjek dan panggil `deleteFunction` metodenya. Buat DeleteFunctionRequestobjek dan berikan ke `deleteFunction` metode. Objek ini berisi informasi seperti nama fungsi yang akan dihapus. Nama fungsi muncul sebagai arn:aws:lambda: us-east-1:555556330391:function:. HelloFunction Anda dapat mengambil nilai dengan melihat fungsi di. Konsol Manajemen AWS

Impor

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.lambda.AWSLambda;
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;
import com.amazonaws.services.lambda.model.ServiceException;
import com.amazonaws.services.lambda.model.DeleteFunctionRequest;
```

Kode

Kode Java berikut menunjukkan cara menghapus Lambda fungsi.

```
        String functionName = args[0];
        try {
            AWSLambda awsLambda = AWSLambdaClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(Regions.US_WEST_2).build();

            DeleteFunctionRequest delFunc = new DeleteFunctionRequest();
            delFunc.withFunctionName(functionName);

            //Delete the function
            awsLambda.deleteFunction(delFunc);
            System.out.println("The function is deleted");

        } catch (ServiceException e) {
            System.out.println(e);
```

```
        }
```

Lihat contoh lengkapnya di Github.

Amazon Pinpoint Contoh Menggunakan AWS SDK untuk Java

Bagian ini memberikan contoh pemrograman Amazon Pinpointmenggunakan AWS SDK untuk Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. Kode contoh lengkap tersedia di GitHub. Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

Membuat dan Menghapus Aplikasi di Amazon Pinpoint

Aplikasi adalah Amazon Pinpoint proyek di mana Anda menentukan audiens untuk aplikasi yang berbeda, dan Anda melibatkan audiens ini dengan pesan yang disesuaikan. Contoh di halaman ini menunjukkan cara membuat aplikasi baru atau menghapus yang sudah ada.

Buat Aplikasi

Buat aplikasi baru Amazon Pinpoint dengan memberikan nama aplikasi ke CreateAppRequestobjek, lalu meneruskan objek AmazonPinpointClient itu ke `createApp` metode.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
```

```
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateAppRequest;
import com.amazonaws.services.pinpoint.model.CreateAppResult;
import com.amazonaws.services.pinpoint.model.CreateApplicationRequest;
```

Kode

```
CreateApplicationRequest appRequest = new CreateApplicationRequest()
  .withName(appName);

CreateAppRequest request = new CreateAppRequest();
request.withCreateApplicationRequest(appRequest);
CreateAppResult result = pinpoint.createApp(request);
```

Lihat contoh lengkapnya di GitHub.

Menghapus Aplikasi

Untuk menghapus aplikasi, panggil AmazonPinpointClient `deleteApp` permintaan dengan DeleteAppRequestobjek yang disetel dengan nama aplikasi yang akan dihapus.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
```

Kode

```
DeleteAppRequest deleteRequest = new DeleteAppRequest()
  .withApplicationId(appID);

pinpoint.deleteApp(deleteRequest);
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- Aplikasi di Referensi Amazon Pinpoint API
- Aplikasi di Referensi Amazon Pinpoint API

Membuat Endpoint di Amazon Pinpoint

Endpoint secara unik mengidentifikasi perangkat pengguna yang dapat Anda gunakan untuk mengirim pemberitahuan push. Amazon Pinpoint Jika aplikasi Anda diaktifkan dengan Amazon Pinpoint dukungan, aplikasi Anda akan secara otomatis mendaftarkan titik akhir Amazon Pinpoint saat pengguna baru membuka aplikasi Anda. Contoh berikut menunjukkan bagaimana menambahkan endpoint baru secara terprogram.

Buat Endpoint

Buat endpoint baru Amazon Pinpoint dengan menyediakan data endpoint dalam sebuah EndpointRequestobjek.

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 buat UpdateEndpointRequestobjek dengan EndpointRequest objek itu. Akhirnya, berikan UpdateEndpointRequest objek ke AmazonPinpointClient `updateEndpoint` metode ini.

Kode

```
UpdateEndpointRequest updateEndpointRequest = new UpdateEndpointRequest()
        .withApplicationId(appId)
        .withEndpointId(endpointId)
        .withEndpointRequest(endpointRequest);
```

```
UpdateEndpointResult updateEndpointResponse =
 client.updateEndpoint(updateEndpointRequest);
System.out.println("Update Endpoint Response: " +
 updateEndpointResponse.getMessageBody());
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- Menambahkan Endpoint di Panduan Amazon Pinpoint Pengembang
- Titik akhir dalam Referensi Amazon Pinpoint API

Membuat Segmen di Amazon Pinpoint

Segmen pengguna mewakili subset pengguna Anda yang didasarkan pada karakteristik bersama, seperti seberapa baru pengguna membuka aplikasi Anda atau perangkat mana yang mereka gunakan. Contoh berikut menunjukkan bagaimana mendefinisikan segmen pengguna.

Buat Segmen

Buat segmen baru Amazon Pinpoint dengan mendefinisikan dimensi segmen dalam suatu SegmentDimensionsobjek.

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);
```

Selanjutnya mengatur SegmentDimensionsobjek dalam a WriteSegmentRequest, yang pada gilirannya digunakan untuk membuat CreateSegmentRequestobjek. Kemudian berikan CreateSegmentRequest objek ke `createSegment` metode AmazonPinpointClient ini.

Kode

```
WriteSegmentRequest writeSegmentRequest = new WriteSegmentRequest()
        .withName("MySegment").withDimensions(dimensions);

CreateSegmentRequest createSegmentRequest = new CreateSegmentRequest()
        .withApplicationId(appId).withWriteSegmentRequest(writeSegmentRequest);

CreateSegmentResult createSegmentResult = client.createSegment(createSegmentRequest);
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- Amazon Pinpoint Segmen dalam Panduan Amazon Pinpoint Pengguna
- Membuat Segmen di Panduan Amazon Pinpoint Pengembang
- Segmen dalam Referensi Amazon Pinpoint API

- Segmen dalam Referensi Amazon Pinpoint API

Membuat Kampanye di Amazon Pinpoint

Anda dapat menggunakan kampanye untuk membantu meningkatkan interaksi antara aplikasi dan pengguna. Anda dapat membuat kampanye untuk menjangkau segmen tertentu dari pengguna Anda dengan pesan yang disesuaikan atau promosi khusus. Contoh ini menunjukkan cara membuat kampanye standar baru yang mengirimkan pemberitahuan push kustom ke segmen tertentu.

Buat 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 baru Amazon Pinpoint WriteCampaignRequestdengan menyediakan konfigurasi kampanye ke CreateCampaignRequestobjek. Akhirnya, berikan CreateCampaignRequest objek ke AmazonPinpointClient `createCampaign` metode ini.

Kode

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

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

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- Amazon Pinpoint Kampanye di Panduan Amazon Pinpoint Pengguna
- Membuat Kampanye di Panduan Amazon Pinpoint Pengembang
- Kampanye di Referensi Amazon Pinpoint API
- Kampanye di Referensi Amazon Pinpoint API
- Aktivitas Kampanye di Referensi Amazon Pinpoint API
- Versi Kampanye di Referensi Amazon Pinpoint API
- Versi Kampanye di Referensi Amazon Pinpoint API

Memperbarui Saluran di Amazon Pinpoint

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

Memperbarui Saluran

Aktifkan saluran Amazon Pinpoint dengan memberikan ID aplikasi dan objek permintaan dari jenis saluran yang ingin Anda perbarui. Contoh ini memperbarui APNs saluran, yang memerlukan

objek APNSChannelRequest. Atur ini di UpdateApnsChannelRequestdan berikan objek AmazonPinpointClient itu ke `updateApnsChannel` metode ini.

Impor

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.APNSChannelRequest;
import com.amazonaws.services.pinpoint.model.APNSChannelResponse;
import com.amazonaws.services.pinpoint.model.GetApnsChannelRequest;
import com.amazonaws.services.pinpoint.model.GetApnsChannelResult;
import com.amazonaws.services.pinpoint.model.UpdateApnsChannelRequest;
import com.amazonaws.services.pinpoint.model.UpdateApnsChannelResult;
```

Kode

```
APNSChannelRequest request = new APNSChannelRequest()
  .withEnabled(enabled);

UpdateApnsChannelRequest updateRequest = new UpdateApnsChannelRequest()
  .withAPNSChannelRequest(request)
  .withApplicationId(appId);
UpdateApnsChannelResult result = client.updateApnsChannel(updateRequest);
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- Amazon Pinpoint Saluran dalam Panduan Amazon Pinpoint Pengguna
- Saluran ADM di Referensi Amazon Pinpoint API
- APNs Saluran di Referensi Amazon Pinpoint API
- APNs Saluran Kotak Pasir di Referensi Amazon Pinpoint API
- APNs Saluran VoIP di Referensi API Amazon Pinpoint
- APNs Saluran Kotak Pasir VoIP di Referensi API Amazon Pinpoint
- Saluran Baidu di Referensi Amazon Pinpoint API
- Saluran Email di Referensi Amazon Pinpoint API
- Saluran GCM di Referensi Amazon Pinpoint API
- Saluran SMS di Referensi Amazon Pinpoint API

Amazon S3 Contoh Menggunakan AWS SDK untuk Java

Bagian ini memberikan contoh pemrograman Amazon S3menggunakan AWS SDK untuk Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. Kode contoh lengkap tersedia di GitHub. Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

Membuat, Membuat Daftar, dan Menghapus Bucket Amazon S3

Setiap objek (file) di Amazon S3 harus berada dalam ember, yang mewakili koleksi (wadah) objek. Setiap ember dikenal dengan kunci (nama), yang harus unik. Untuk informasi terperinci tentang bucket dan konfigurasinya, lihat Bekerja dengan Amazon S3 Bucket di Amazon Simple Storage Service Panduan Pengguna.

> **Note**
>
> Praktik Terbaik
> Kami menyarankan Anda mengaktifkan aturan AbortIncompleteMultipartUploadsiklus hidup pada bucket Anda Amazon S3 .
> Aturan ini mengarahkan Amazon S3 untuk membatalkan unggahan multipart yang tidak selesai dalam jumlah hari tertentu setelah dimulai. Ketika batas waktu yang ditetapkan

terlampaui, Amazon S3 batalkan unggahan dan kemudian menghapus data unggahan yang tidak lengkap.
Untuk informasi selengkapnya, lihat Konfigurasi Siklus Hidup untuk Bucket dengan Pembuatan Versi di Panduan Pengguna. Amazon S3

> **Note**
>
> Contoh kode ini mengasumsikan bahwa Anda memahami materi dalam Menggunakan AWS SDK untuk Java dan telah mengonfigurasi AWS kredensi default menggunakan informasi di Siapkan AWS Kredensial dan Wilayah untuk Pengembangan.

Buat Bucket

Gunakan metode klien AmazonS3. `createBucket` Bucket baru dikembalikan. `createBucket`Metode ini akan memunculkan pengecualian jika bucket sudah ada.

> **Note**
>
> Untuk memeriksa apakah bucket sudah ada sebelum mencoba membuatnya dengan nama yang sama, panggil `doesBucketExist` metode tersebut. Ini akan kembali `true` jika ember ada, dan `false` sebaliknya.

Impor

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;
```

Kode

```
if (s3.doesBucketExistV2(bucket_name)) {
```

```
    System.out.format("Bucket %s already exists.\n", bucket_name);
    b = getBucket(bucket_name);
} else {
    try {
        b = s3.createBucket(bucket_name);
    } catch (AmazonS3Exception e) {
        System.err.println(e.getErrorMessage());
    }
}
return b;
```

Lihat contoh lengkapnya di GitHub.

Buat Daftar Bucket

Gunakan metode klien AmazonS3. `listBucket` Jika berhasil, daftar Bucket dikembalikan.

Impor

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;
```

Kode

```
List<Bucket> buckets = s3.listBuckets();
System.out.println("Your {S3} buckets are:");
for (Bucket b : buckets) {
    System.out.println("* " + b.getName());
}
```

Lihat contoh lengkapnya di GitHub.

Hapus Bucket

Sebelum Anda dapat menghapus Amazon S3 ember, Anda harus memastikan bahwa ember kosong atau kesalahan akan terjadi. Jika Anda memiliki bucket berversi, Anda juga harus menghapus objek berversi apa pun yang terkait dengan bucket.

> **Note**
>
> Contoh lengkap mencakup masing-masing langkah ini secara berurutan, memberikan solusi lengkap untuk menghapus Amazon S3 ember dan isinya.

Topik

- Hapus Objek dari Bucket Tidak Berversi Sebelum Menghapusnya
- Hapus Objek dari Bucket Berversi Sebelum Menghapusnya
- Hapus Ember Kosong

Hapus Objek dari Bucket Tidak Berversi Sebelum Menghapusnya

Gunakan `listObjects` metode klien AmazonS3 untuk mengambil daftar objek dan `deleteObject` menghapus masing-masing 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.*;

import java.util.Iterator;
```

Kode

```
System.out.println(" - removing objects from bucket");
ObjectListing object_listing = s3.listObjects(bucket_name);
while (true) {
    for (Iterator<?> iterator =
         object_listing.getObjectSummaries().iterator();
         iterator.hasNext(); ) {
        S3ObjectSummary summary = (S3ObjectSummary) iterator.next();
        s3.deleteObject(bucket_name, summary.getKey());
    }

    // more object_listing to retrieve?
    if (object_listing.isTruncated()) {
```

```
        object_listing = s3.listNextBatchOfObjects(object_listing);
    } else {
        break;
    }
}
```

Lihat contoh lengkapnya di GitHub.

Hapus Objek dari Bucket Berversi Sebelum Menghapusnya

Jika Anda menggunakan bucket berversi, Anda juga perlu menghapus semua versi objek yang tersimpan di bucket sebelum bucket dapat dihapus.

Menggunakan pola yang mirip dengan yang digunakan saat menghapus objek di dalam ember, hapus objek berversi dengan menggunakan `listVersions` metode klien AmazonS3 untuk mencantumkan objek berversi apa pun, dan kemudian menghapus masing-masing objek. `deleteVersion`

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;
```

Kode

```
System.out.println(" - removing versions from bucket");
VersionListing version_listing = s3.listVersions(
        new ListVersionsRequest().withBucketName(bucket_name));
while (true) {
    for (Iterator<?> iterator =
         version_listing.getVersionSummaries().iterator();
         iterator.hasNext(); ) {
        S3VersionSummary vs = (S3VersionSummary) iterator.next();
        s3.deleteVersion(
                bucket_name, vs.getKey(), vs.getVersionId());
    }
```

```
    if (version_listing.isTruncated()) {
        version_listing = s3.listNextBatchOfVersions(
                version_listing);
    } else {
        break;
    }
}
```

Lihat contoh lengkapnya di GitHub.

Hapus Ember Kosong

Setelah Anda menghapus objek dari bucket (termasuk objek berversi apa pun), Anda dapat menghapus bucket itu sendiri dengan menggunakan metode klien AmazonS3. `deleteBucket`

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;
```

Kode

```
System.out.println(" OK, bucket ready to delete!");
s3.deleteBucket(bucket_name);
```

Lihat contoh lengkapnya di GitHub.

Melakukan Operasi pada Amazon S3 Objek

Amazon S3 Objek mewakili file atau kumpulan data. Setiap benda harus berada di dalam ember.

> **Note**
>
> Contoh kode ini mengasumsikan bahwa Anda memahami materi dalam Menggunakan AWS SDK untuk Java dan telah mengonfigurasi AWS kredensyal default menggunakan informasi di Siapkan AWS Kredensyal dan Wilayah untuk Pengembangan.

Topik

Meng-unggah Objek

Gunakan `putObject` metode klien AmazonS3, berikan nama bucket, nama kunci, dan file untuk diunggah. Bucket harus ada, atau kesalahan 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 lengkapnya di GitHub.

Daftar Objek

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

`listObjects`Metode mengembalikan ObjectListingobjek yang memberikan informasi tentang objek dalam ember. Untuk membuat daftar nama objek (kunci), gunakan `getObjectSummaries` metode untuk mendapatkan Daftar ObjectSummary objek S3, yang masing-masing mewakili satu objek dalam ember. Kemudian panggil `getKey` metodenya untuk mengambil nama objek.

Impor

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;
```

Kode

```
System.out.format("Objects in S3 bucket %s:\n", bucket_name);
final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
ListObjectsV2Result result = s3.listObjectsV2(bucket_name);
List<S3ObjectSummary> objects = result.getObjectSummaries();
for (S3ObjectSummary os : objects) {
    System.out.println("* " + os.getKey());
}
```

Lihat contoh lengkapnya di GitHub.

Mengunduh Objek

Gunakan `getObject` metode klien AmazonS3, berikan nama bucket dan objek untuk diunduh. Jika berhasil, metode mengembalikan S3Object. Bucket dan kunci objek yang ditentukan harus ada, atau kesalahan akan terjadi.

Anda bisa mendapatkan konten objek dengan memanggil `getObjectContentS3Object`. Ini mengembalikan S3 ObjectInputStream yang berperilaku sebagai objek Java `InputStream` standar.

Contoh berikut mengunduh objek dari S3 dan menyimpan isinya ke file (menggunakan nama yang sama dengan kunci objek).

Impor

```
import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import java.io.File;
```

Kode

```
System.out.format("Downloading %s from S3 bucket %s...\n", key_name, bucket_name);
final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    S3Object o = s3.getObject(bucket_name, key_name);
    S3ObjectInputStream s3is = o.getObjectContent();
    FileOutputStream fos = new FileOutputStream(new File(key_name));
    byte[] read_buf = new byte[1024];
    int read_len = 0;
    while ((read_len = s3is.read(read_buf)) > 0) {
        fos.write(read_buf, 0, read_len);
    }
    s3is.close();
    fos.close();
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
} catch (FileNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Lihat contoh lengkapnya di GitHub.

Salin, Pindahkan, atau Ganti Nama Objek

Anda dapat menyalin objek dari satu bucket ke bucket lainnya dengan menggunakan metode klien AmazonS3. `copyObject` Dibutuhkan nama bucket untuk disalin, objek yang akan disalin, 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 lengkapnya di GitHub.

> **Note**
>
> Anda dapat menggunakan `copyObject` dengan DeleteObject untuk memindahkan atau mengganti nama objek, dengan terlebih dahulu menyalin objek ke nama baru (Anda dapat menggunakan bucket yang sama dengan 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 kesalahan 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 lengkapnya di GitHub.

Hapus Beberapa Objek Sekaligus

Menggunakan `deleteObjects` metode klien AmazonS3, Anda dapat menghapus beberapa objek dari bucket yang sama dengan meneruskan nama mereka ke metode link:sdk-for-java/v1/reference/com/amazonaws/services/s3/model/DeleteObjectsRequest.html.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
```

Kode

```
final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    DeleteObjectsRequest dor = new DeleteObjectsRequest(bucket_name)
            .withKeys(object_keys);
    s3.deleteObjects(dor);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Lihat contoh lengkapnya di GitHub.

Mengelola Izin Amazon S3 Akses untuk Bucket dan Objek

Anda dapat menggunakan daftar kontrol akses (ACLs) untuk Amazon S3 bucket dan objek untuk kontrol halus atas sumber daya Anda. Amazon S3

> **Note**
>
> Contoh kode ini mengasumsikan bahwa Anda memahami materi dalam Menggunakan AWS SDK untuk Java dan telah mengonfigurasi AWS kredensi default menggunakan informasi di Siapkan AWS Kredensial dan Wilayah untuk Pengembangan.

Dapatkan Daftar Kontrol Akses untuk Bucket

Untuk mendapatkan ACL saat ini untuk bucket, panggil `getBucketAcl` metode AmazonS3, berikan nama bucket ke kueri. Metode ini mengembalikan sebuah AccessControlListobjek. Untuk mendapatkan setiap hibah akses dalam daftar, panggil `getGrantsAsList` metodenya, yang akan mengembalikan daftar Java standar objek Grant.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.Grant;
```

Kode

```
final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    AccessControlList acl = s3.getBucketAcl(bucket_name);
    List<Grant> grants = acl.getGrantsAsList();
    for (Grant grant : grants) {
        System.out.format("  %s: %s\n", grant.getGrantee().getIdentifier(),
                grant.getPermission().toString());
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Lihat contoh lengkapnya di GitHub.

Mengatur Daftar Kontrol Akses untuk Bucket

Untuk menambahkan atau memodifikasi izin ke ACL untuk bucket, panggil metode AmazonS3. `setBucketAcl` Dibutuhkan AccessControlListobjek yang berisi daftar penerima hibah dan tingkat akses untuk ditetapkan.

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 menggunakan kelas Penerima, atau menggunakan EmailAddressGranteekelas untuk mengatur penerima hibah melalui email, seperti yang telah kita lakukan di sini.

Lihat contoh lengkapnya di GitHub.

Dapatkan Daftar Kontrol Akses untuk Objek

Untuk mendapatkan ACL saat ini untuk suatu objek, panggil `getObjectAcl` metode AmazonS3, berikan nama bucket dan nama objek ke kueri. Seperti`getBucketAcl`, metode ini mengembalikan AccessControlListobjek yang dapat Anda gunakan untuk memeriksa setiap Grant.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.Grant;
```

Kode

```
try {
    AccessControlList acl = s3.getObjectAcl(bucket_name, object_key);
    List<Grant> grants = acl.getGrantsAsList();
    for (Grant grant : grants) {
        System.out.format("  %s: %s\n", grant.getGrantee().getIdentifier(),
                grant.getPermission().toString());
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Lihat contoh lengkapnya di GitHub.

Mengatur Daftar Kontrol Akses untuk Objek

Untuk menambahkan atau memodifikasi izin ke ACL untuk objek, panggil metode AmazonS3. `setObjectAcl` Dibutuhkan AccessControlListobjek yang berisi daftar penerima hibah dan tingkat akses untuk ditetapkan.

Impor

```
import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
```

Kode

```
    try {
        // get the current ACL
        AccessControlList acl = s3.getObjectAcl(bucket_name, object_key);
        // set access for the grantee
        EmailAddressGrantee grantee = new EmailAddressGrantee(email);
        Permission permission = Permission.valueOf(access);
        acl.grantPermission(grantee, permission);
        s3.setObjectAcl(bucket_name, object_key, acl);
    } catch (AmazonServiceException e) {
        System.err.println(e.getErrorMessage());
        System.exit(1);
    }
}
```

> **Note**
>
> Anda dapat memberikan pengenal unik penerima hibah secara langsung menggunakan kelas Penerima, atau menggunakan EmailAddressGranteekelas untuk mengatur penerima hibah melalui email, seperti yang telah kita lakukan di sini.

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- DAPATKAN Bucket acl di Referensi Amazon S3 API
- PUT Bucket acl di Referensi Amazon S3 API
- GET Object acl di Referensi Amazon S3 API
- PUT Object acl di Referensi Amazon S3 API

Mengelola Akses ke Amazon S3 Bucket Menggunakan Kebijakan Bucket

Anda dapat menyetel, mendapatkan, atau menghapus kebijakan bucket untuk mengelola akses ke Amazon S3 bucket Anda.

Menetapkan Kebijakan Bucket

Anda dapat menyetel kebijakan bucket untuk bucket S3 tertentu dengan:

- Memanggil klien AmazonS3 `setBucketPolicy` dan menyediakannya SetBucketPolicyRequest
- Menyetel kebijakan secara langsung dengan menggunakan `setBucketPolicy` overload 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);
}
```

Menggunakan Kelas Kebijakan untuk Menghasilkan atau Memvalidasi Kebijakan

Saat memberikan kebijakan bucket`setBucketPolicy`, Anda dapat melakukan hal berikut:

- Tentukan kebijakan secara langsung sebagai string teks berformat JSON
- Membangun kebijakan menggunakan class Policy

Dengan menggunakan `Policy` kelas, Anda tidak perlu khawatir tentang memformat string teks Anda dengan benar. Untuk mendapatkan teks kebijakan JSON dari `Policy` kelas, gunakan `toJson` metodenya.

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();
```

`Policy`Kelas juga menyediakan `fromJson` metode yang dapat mencoba untuk membangun kebijakan menggunakan string JSON passed-in. Metode memvalidasinya untuk memastikan bahwa teks dapat diubah menjadi struktur kebijakan yang valid, dan akan gagal dengan `IllegalArgumentException` jika teks kebijakan tidak valid.

```
Policy bucket_policy = null;
try {
    bucket_policy = Policy.fromJson(file_text.toString());
} catch (IllegalArgumentException e) {
    System.out.format("Invalid policy text in file: \"%s\"",
            policy_file);
    System.out.println(e.getMessage());
}
```

Anda dapat menggunakan teknik ini untuk mencegah kebijakan yang Anda baca dari file atau cara lain.

Lihat contoh lengkapnya di GitHub.

Dapatkan Kebijakan Bucket

Untuk mengambil kebijakan Amazon S3 bucket, panggil `getBucketPolicy` metode klien AmazonS3, berikan nama bucket untuk mendapatkan kebijakan tersebut.

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, maka akan `AmazonServiceException` dilemparkan.

Lihat contoh lengkapnya di GitHub.

Menghapus Kebijakan Bucket

Untuk menghapus kebijakan bucket, hubungi klien AmazonS3`deleteBucketPolicy`, berikan 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, maka akan `AmazonServiceException` ditampilkan.

Lihat contoh lengkapnya di GitHub.

Info Selengkapnya

- Ikhtisar Bahasa Kebijakan Akses di Panduan Amazon Simple Storage Service Pengguna
- Contoh Kebijakan Bucket di Panduan Amazon Simple Storage Service Pengguna

Menggunakan TransferManager untuk Amazon S3 Operasi

Anda dapat menggunakan AWS SDK untuk Java TransferManager kelas untuk mentransfer file secara andal dari lingkungan lokal ke Amazon S3 dan untuk menyalin objek dari satu lokasi S3 ke lokasi lainnya. `TransferManager`bisa mendapatkan kemajuan transfer dan jeda atau melanjutkan unggahan dan unduhan.

> **Note**
>
> Praktik Terbaik
> Kami menyarankan Anda mengaktifkan aturan AbortIncompleteMultipartUploadsiklus hidup pada bucket Anda Amazon S3 .
> Aturan ini mengarahkan Amazon S3 untuk membatalkan unggahan multipart yang tidak selesai dalam jumlah hari tertentu setelah dimulai. Ketika batas waktu yang ditetapkan terlampaui, Amazon S3 batalkan unggahan dan kemudian menghapus data unggahan yang tidak lengkap.
> Untuk informasi selengkapnya, lihat Konfigurasi Siklus Hidup untuk Bucket dengan Pembuatan Versi di Panduan Pengguna. Amazon S3

> **Note**
>
> Contoh kode ini mengasumsikan bahwa Anda memahami materi dalam Menggunakan AWS SDK untuk Java dan telah mengonfigurasi AWS kredensi default menggunakan informasi di Siapkan AWS Kredensial dan Wilayah untuk Pengembangan.

Unggah File dan Direktori

TransferManager dapat mengunggah file, daftar file, dan direktori ke Amazon S3 bucket apa pun yang telah Anda buat sebelumnya.

Topik

- Unggah Satu File
- Unggah Daftar File
- Unggah Direktori

Unggah Satu File

`upload`Metode panggilan TransferManager, memberikan nama Amazon S3 bucket, nama kunci (objek), dan objek Java File standar yang mewakili file yang akan 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

```
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();
```

`upload`Metode segera kembali, menyediakan `Upload` objek yang akan digunakan untuk memeriksa status transfer atau menunggu sampai selesai.

Lihat Tunggu Transfer Selesai untuk informasi tentang penggunaan `waitForCompletion` agar berhasil menyelesaikan transfer sebelum `shutdownNow` metode panggilan TransferManager. Sambil menunggu transfer selesai, Anda dapat melakukan polling atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat Dapatkan Status Transfer dan Kemajuan untuk informasi selengkapnya.

Lihat contoh lengkapnya di GitHub.

Unggah Daftar File

Untuk mengunggah beberapa file dalam satu operasi, panggil TransferManager `uploadFileList` metode, berikan yang berikut:

- Nama Amazon S3 ember
- Sebuah key prefix untuk menambahkan ke nama-nama objek yang dibuat (jalur di dalam ember untuk menempatkan objek)
- Sebuah objek File yang mewakili direktori relatif dari mana untuk membuat jalur file
- Sebuah objek List yang berisi satu set objek File untuk meng-upload

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();
```

Lihat Tunggu Transfer Selesai untuk informasi tentang penggunaan `waitForCompletion` agar berhasil menyelesaikan transfer sebelum `shutdownNow` metode panggilan TransferManager. Sambil menunggu transfer selesai, Anda dapat melakukan polling atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat Dapatkan Status Transfer dan Kemajuan untuk informasi selengkapnya.

MultipleFileUploadObjek yang dikembalikan oleh `uploadFileList` dapat digunakan untuk menanyakan status transfer atau kemajuan. Lihat Polling Kemajuan Transfer Saat Ini dan Dapatkan Kemajuan Transfer dengan a ProgressListener untuk informasi selengkapnya.

Anda juga dapat menggunakan `MultipleFileUpload` `getSubTransfers` metode ini untuk mendapatkan `Upload` objek individual untuk setiap file yang ditransfer. Untuk informasi selengkapnya, lihat Mendapatkan Kemajuan Subtransfer.

Lihat contoh lengkapnya di GitHub.

Unggah Direktori

Anda dapat menggunakan TransferManager `uploadDirectory` metode ini untuk mengunggah seluruh direktori file, dengan opsi untuk menyalin file di subdirektori secara rekursif. Anda memberikan nama Amazon S3 bucket, key prefix S3, objek File yang mewakili direktori lokal yang akan disalin, dan `boolean` nilai yang menunjukkan apakah Anda ingin menyalin subdirektori secara rekursif (benar atau salah).

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.MultipleFileUpload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
```

Kode

```
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    MultipleFileUpload xfer = xfer_mgr.uploadDirectory(bucket_name,
            key_prefix, new File(dir_path), recursive);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Lihat Tunggu Transfer Selesai untuk informasi tentang penggunaan `waitForCompletion` agar berhasil menyelesaikan transfer sebelum `shutdownNow` metode panggilan TransferManager. Sambil menunggu transfer selesai, Anda dapat melakukan polling atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat Dapatkan Status Transfer dan Kemajuan untuk informasi selengkapnya.

MultipleFileUploadObjek yang dikembalikan oleh `uploadFileList` dapat digunakan untuk menanyakan status transfer atau kemajuan. Lihat Polling Kemajuan Transfer Saat Ini dan Dapatkan Kemajuan Transfer dengan a ProgressListener untuk informasi selengkapnya.

Anda juga dapat menggunakan `MultipleFileUpload` `getSubTransfers` metode ini untuk mendapatkan `Upload` objek individual untuk setiap file yang ditransfer. Untuk informasi selengkapnya, lihat Mendapatkan Kemajuan Subtransfer.

Lihat contoh lengkapnya di GitHub.

Unduh File atau Direktori

Gunakan TransferManager kelas untuk mengunduh file tunggal (Amazon S3 objek) atau direktori (nama Amazon S3 bucket diikuti oleh awalan objek) dari Amazon S3.

Topik

Unduh File Tunggal

Gunakan `download` metode ini, berikan nama Amazon S3 bucket yang berisi objek yang ingin Anda unduh, nama kunci (objek), dan objek File yang mewakili file yang akan dibuat di sistem lokal Anda. TransferManager

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();
```

Lihat Tunggu Transfer Selesai untuk informasi tentang penggunaan `waitForCompletion` agar berhasil menyelesaikan transfer sebelum `shutdownNow` metode panggilan TransferManager. Sambil menunggu transfer selesai, Anda dapat melakukan polling atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat Dapatkan Status Transfer dan Kemajuan untuk informasi selengkapnya.

Lihat contoh lengkapnya di GitHub.

Unduh Direktori

Untuk mengunduh sekumpulan file yang berbagi key prefix umum (analog dengan direktori pada sistem file) dari Amazon S3, gunakan metode ini. TransferManager `downloadDirectory` Metode ini mengambil nama Amazon S3 bucket yang berisi objek yang ingin Anda unduh, awalan objek yang dibagikan oleh semua objek, dan objek File yang mewakili direktori untuk mengunduh file di sistem lokal Anda. Jika direktori bernama belum ada, itu 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();
```

Lihat Tunggu Transfer Selesai untuk informasi tentang penggunaan `waitForCompletion` agar berhasil menyelesaikan transfer sebelum `shutdownNow` metode panggilan TransferManager. Sambil menunggu transfer selesai, Anda dapat melakukan polling atau mendengarkan pembaruan tentang status dan kemajuannya. Lihat Dapatkan Status Transfer dan Kemajuan untuk informasi selengkapnya.

Lihat contoh lengkapnya di GitHub.

Menyalin objek

Untuk menyalin objek dari satu bucket S3 ke bucket lainnya, gunakan TransferManager `copy` metode ini.

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();
```

Lihat contoh lengkapnya di GitHub.

Tunggu Transfer Selesai

Jika aplikasi Anda (atau thread) dapat memblokir hingga transfer selesai, Anda dapat menggunakan `waitForCompletion` metode antarmuka Transfer untuk memblokir hingga 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 melakukan polling untuk acara sebelum meneleponwaitForCompletion, menerapkan mekanisme polling pada utas terpisah, atau menerima pembaruan kemajuan secara asinkron menggunakan file. ProgressListener

Lihat contoh lengkapnya di GitHub.

Dapatkan Status Transfer dan Kemajuan

Masing-masing kelas dikembalikan oleh TransferManagerupload*,download*, dan copy metode mengembalikan instance dari salah satu kelas berikut, tergantung pada apakah itu operasi satu file atau beberapa file.

Kelas	Dikembalikan oleh
Salin	`copy`
Unduh	`download`
MultipleFileDownload	`downloadDirectory`

Kelas	Dikembalikan oleh
Unggah	`upload`
MultipleFileUpload	`uploadFileList` , `uploadDirectory`

Semua kelas ini menerapkan antarmuka Transfer. `Transfer`menyediakan metode yang berguna untuk mendapatkan kemajuan transfer, menjeda atau melanjutkan transfer, dan mendapatkan status transfer saat ini atau akhir.

Topik

- Polling Kemajuan Transfer Saat Ini
- Dapatkan Kemajuan Transfer dengan ProgressListener
- Dapatkan Kemajuan Subtransfer

Polling Kemajuan Transfer Saat Ini

Loop ini mencetak kemajuan transfer, memeriksa kemajuannya saat berjalan dan, ketika selesai, mencetak status akhirnya.

Impor

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;

import java.io.File;
import java.util.ArrayList;
import java.util.Collection;
```

Kode

```
// print the transfer's human-readable description
System.out.println(xfer.getDescription());
// print an empty progress bar...
printProgressBar(0.0);
```

```
// update the progress bar while the xfer is ongoing.
do {
    try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        return;
    }
    // Note: so_far and total aren't used, they're just for
    // documentation purposes.
    TransferProgress progress = xfer.getProgress();
    long so_far = progress.getBytesTransferred();
    long total = progress.getTotalBytesToTransfer();
    double pct = progress.getPercentTransferred();
    eraseProgressBar();
    printProgressBar(pct);
} while (xfer.isDone() == false);
// print the final state of the transfer.
TransferState xfer_state = xfer.getState();
System.out.println(": " + xfer_state);
```

Lihat contoh lengkapnya di GitHub.

Dapatkan Kemajuan Transfer dengan ProgressListener

Anda dapat melampirkan ProgressListenerke transfer apa pun dengan menggunakan `addProgressListener` metode antarmuka Transfer.

A hanya ProgressListenermembutuhkan satu metode`progressChanged`, yang mengambil ProgressEventobjek. Anda dapat menggunakan objek untuk mendapatkan total byte operasi dengan memanggil `getBytes` metodenya, dan jumlah byte yang ditransfer sejauh ini dengan memanggil. `getBytesTransferred`

Impor

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;

import java.io.File;
import java.util.ArrayList;
```

```
import java.util.Collection;
```

Kode

```
File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Upload u = xfer_mgr.upload(bucket_name, key_name, f);
    // print an empty progress bar...
    printProgressBar(0.0);
    u.addProgressListener(new ProgressListener() {
        public void progressChanged(ProgressEvent e) {
            double pct = e.getBytesTransferred() * 100.0 / e.getBytes();
            eraseProgressBar();
            printProgressBar(pct);
        }
    });
    // block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(u);
    // print the final state of the transfer.
    TransferState xfer_state = u.getState();
    System.out.println(": " + xfer_state);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Lihat contoh lengkapnya di GitHub.

Dapatkan Kemajuan Subtransfer

MultipleFileUploadKelas dapat mengembalikan informasi tentang subtransfernya dengan memanggil `getSubTransfers` metodenya. Ini mengembalikan Koleksi objek Unggah yang tidak dapat dimodifikasi yang menyediakan status transfer individu dan kemajuan setiap sub-transfer.

Impor

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
```

```
import com.amazonaws.services.s3.transfer.*;
import com.amazonaws.services.s3.transfer.Transfer.TransferState;

import java.io.File;
import java.util.ArrayList;
import java.util.Collection;
```

Kode

```
Collection<? extends Upload> sub_xfers = new ArrayList<Upload>();
sub_xfers = multi_upload.getSubTransfers();

do {
    System.out.println("\nSubtransfer progress:\n");
    for (Upload u : sub_xfers) {
        System.out.println("  " + u.getDescription());
        if (u.isDone()) {
            TransferState xfer_state = u.getState();
            System.out.println("  " + xfer_state);
        } else {
            TransferProgress progress = u.getProgress();
            double pct = progress.getPercentTransferred();
            printProgressBar(pct);
            System.out.println();
        }
    }

    // wait a bit before the next update.
    try {
        Thread.sleep(200);
    } catch (InterruptedException e) {
        return;
    }
} while (multi_upload.isDone() == false);
// print the final state of the transfer.
TransferState xfer_state = multi_upload.getState();
System.out.println("\nMultipleFileUpload " + xfer_state);
```

Lihat contoh lengkapnya di GitHub.

Info Selengkapnya

- Kunci Objek di Panduan Amazon Simple Storage Service Pengguna

Mengkonfigurasi Amazon S3 Bucket sebagai Situs Web

Anda dapat mengonfigurasi Amazon S3 bucket untuk berperilaku sebagai situs web. Untuk melakukan ini, Anda perlu mengatur konfigurasi situs webnya.

> **Note**
>
> Contoh kode ini mengasumsikan bahwa Anda memahami materi dalam Menggunakan AWS SDK untuk Java dan telah mengonfigurasi AWS kredensi default menggunakan informasi di Siapkan AWS Kredensial dan Wilayah untuk Pengembangan.

Mengatur Konfigurasi Situs Web Bucket

Untuk mengatur konfigurasi situs web Amazon S3 bucket, panggil `setWebsiteConfiguration` metode AmazonS3 dengan nama bucket untuk menyetel konfigurasi, dan BucketWebsiteConfigurationobjek yang berisi konfigurasi situs web bucket.

Diperlukan pengaturan dokumen indeks; 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**
>
> Menyetel konfigurasi situs web tidak mengubah izin akses untuk bucket Anda. Untuk membuat file Anda terlihat di web, Anda juga perlu menetapkan kebijakan bucket yang memungkinkan akses baca publik ke file di bucket. Untuk informasi selengkapnya, lihat Mengelola Akses ke Amazon S3 Bucket Menggunakan Kebijakan Bucket.

Lihat contoh lengkapnya di GitHub.

Dapatkan Konfigurasi Situs Web Bucket

Untuk mendapatkan konfigurasi situs web Amazon S3 bucket, panggil `getWebsiteConfiguration` metode AmazonS3 dengan nama bucket untuk mengambil konfigurasi.

Konfigurasi akan dikembalikan sebagai BucketWebsiteConfigurationobjek. Jika tidak ada konfigurasi situs web untuk bucket, maka `null` akan dikembalikan.

Impor

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketWebsiteConfiguration;
```

Kode

```
final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    BucketWebsiteConfiguration config =
            s3.getBucketWebsiteConfiguration(bucket_name);
    if (config == null) {
        System.out.println("No website configuration found!");
    } else {
        System.out.format("Index document: %s\n",
                config.getIndexDocumentSuffix());
        System.out.format("Error document: %s\n",
                config.getErrorDocument());
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.out.println("Failed to get website configuration!");
    System.exit(1);
}
```

Lihat contoh lengkapnya di GitHub.

Menghapus Konfigurasi Situs Web Bucket

Untuk menghapus konfigurasi situs web Amazon S3 bucket, panggil `deleteWebsiteConfiguration` metode AmazonS3 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);
}
```

Lihat contoh lengkapnya di GitHub.

Informasi Selengkapnya

- Situs web PUT Bucket di Referensi Amazon S3 API
- Dapatkan situs web Bucket di Referensi Amazon S3 API
- HAPUS situs web Bucket di Referensi Amazon S3 API

Gunakan Amazon S3 enkripsi sisi klien

Mengenkripsi data menggunakan klien Amazon S3 enkripsi adalah salah satu cara Anda dapat memberikan lapisan perlindungan tambahan untuk informasi sensitif yang Anda simpan. Amazon S3 Contoh di bagian ini menunjukkan cara membuat dan mengkonfigurasi klien Amazon S3 enkripsi untuk aplikasi Anda.

Jika Anda baru mengenal kriptografi, lihat Dasar-Dasar Kriptografi di Panduan Pengembang AWS KMS untuk ikhtisar dasar istilah dan algoritma kriptografi. Untuk informasi tentang dukungan kriptografi di semua AWS SDKs, lihat AWS Dukungan SDK untuk Enkripsi Amazon S3 Sisi Klien di Referensi Umum. Amazon Web Services

> **Note**
>
> Contoh kode ini mengasumsikan bahwa Anda memahami materi dalam Menggunakan AWS SDK untuk Java dan telah mengonfigurasi AWS kredensi default menggunakan informasi di Siapkan AWS Kredensial dan Wilayah untuk Pengembangan.

Jika Anda menggunakan versi 1.11.836 atau versi sebelumnya AWS SDK untuk Java, lihat Migrasi Klien Amazon S3 Enkripsi untuk informasi tentang memigrasi aplikasi Anda ke versi yang lebih baru. Jika Anda tidak dapat bermigrasi, lihat contoh lengkap ini di GitHub.

Jika tidak, jika Anda menggunakan versi 1.11.837 atau yang lebih baru AWS SDK untuk Java, jelajahi contoh topik yang tercantum di bawah ini untuk menggunakan Amazon S3 enkripsi sisi klien.

Topik

- Amazon S3 enkripsi sisi klien dengan kunci master klien
- Amazon S3 enkripsi sisi klien dengan kunci terkelola AWS KMS

Amazon S3 enkripsi sisi klien dengan kunci master klien

Contoh berikut menggunakan kelas AmazonS3 EncryptionClient V2Builder untuk membuat klien dengan enkripsi sisi Amazon S3 klien diaktifkan. Setelah diaktifkan, objek apa pun yang Anda unggah untuk Amazon S3 menggunakan klien ini akan dienkripsi. Objek apa pun yang Anda dapatkan dari Amazon S3 menggunakan klien ini akan didekripsi secara otomatis.

> **Note**
>
> Contoh berikut menunjukkan penggunaan enkripsi Amazon S3 sisi klien dengan kunci master klien yang dikelola pelanggan. Untuk mempelajari cara menggunakan enkripsi dengan kunci terkelola AWS KMS, lihat enkripsi Amazon S3 sisi klien dengan kunci terkelola AWS KMS.

Anda dapat memilih dari dua mode enkripsi saat mengaktifkan Amazon S3 enkripsi sisi klien: otentikasi ketat atau diautentikasi. Bagian berikut menunjukkan cara mengaktifkan setiap jenis. Untuk mempelajari algoritma mana yang digunakan setiap mode, lihat CryptoModedefinisinya.

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 otentikasi yang ketat

Enkripsi otentikasi yang ketat adalah mode default jika tidak `CryptoMode` ditentukan.

Untuk mengaktifkan mode ini secara eksplisit, tentukan `StrictAuthenticatedEncryption` nilai dalam metode. `withCryptoConfiguration`

> **Note**
>
> Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan file jar Bouncy Castle terbaru 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 yang diautentikasi

Saat Anda menggunakan `AuthenticatedEncryption` mode, algoritma pembungkus kunci yang ditingkatkan diterapkan selama enkripsi. Saat mendekripsi dalam mode ini, algoritme dapat memverifikasi integritas objek yang didekripsi dan melempar pengecualian jika pemeriksaan gagal. Untuk detail selengkapnya tentang cara kerja enkripsi yang diautentikasi, lihat posting blog Enkripsi Terautentikasi Amazon S3 Sisi Klien.

> **Note**
>
> Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan file jar Bouncy Castle terbaru di classpath aplikasi Anda.

Untuk mengaktifkan mode ini, tentukan `AuthenticatedEncryption` nilai 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 S3 enkripsi sisi klien dengan kunci terkelola AWS KMS

Contoh berikut menggunakan kelas AmazonS3 EncryptionClient V2Builder untuk membuat klien dengan enkripsi sisi Amazon S3 klien diaktifkan. Setelah dikonfigurasi, objek apa pun yang Anda unggah untuk Amazon S3 menggunakan klien ini akan dienkripsi. Setiap objek yang Anda dapatkan dari Amazon S3 menggunakan klien ini secara otomatis didekripsi.

> **Note**
>
> Contoh berikut menunjukkan cara menggunakan enkripsi Amazon S3 sisi klien dengan kunci terkelola AWS KMS. Untuk mempelajari cara menggunakan enkripsi dengan kunci Anda sendiri, lihat enkripsi Amazon S3 sisi klien dengan kunci master klien.

Anda dapat memilih dari dua mode enkripsi saat mengaktifkan Amazon S3 enkripsi sisi klien: otentikasi ketat atau diautentikasi. Bagian berikut menunjukkan cara mengaktifkan setiap jenis. Untuk mempelajari algoritma mana yang digunakan setiap mode, lihat CryptoModedefinisinya.

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 otentikasi yang ketat

Enkripsi otentikasi ketat adalah mode default jika tidak `CryptoMode` ditentukan.

Untuk mengaktifkan mode ini secara eksplisit, tentukan `StrictAuthenticatedEncryption` nilai dalam metode. `withCryptoConfiguration`

> **Note**
>
> Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan file jar Bouncy Castle terbaru 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 klien Amazon S3 enkripsi untuk mengunggah 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 yang diautentikasi

Saat Anda menggunakan `AuthenticatedEncryption` mode, algoritma pembungkus kunci yang ditingkatkan diterapkan selama enkripsi. Saat mendekripsi dalam mode ini, algoritme dapat memverifikasi integritas objek yang didekripsi dan melempar pengecualian jika pemeriksaan gagal. Untuk detail selengkapnya tentang cara kerja enkripsi yang diautentikasi, lihat posting blog Enkripsi Terautentikasi Amazon S3 Sisi Klien.

> **Note**
>
> Untuk menggunakan enkripsi otentikasi sisi klien, Anda harus menyertakan file jar Bouncy Castle terbaru di classpath aplikasi Anda.

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

Kode

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

Mengkonfigurasi klien AWS KMS

Klien Amazon S3 enkripsi membuat AWS KMS klien secara default, kecuali satu ditentukan secara eksplisit.

Untuk mengatur wilayah untuk klien yang dibuat secara otomatis ini, atur AWS KMS . `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 AWS KMS klien Anda sendiri untuk menginisialisasi klien enkripsi.

Kode

```
AWSKMS kmsClient = AWSKMSClientBuilder.standard()
        .withRegion(Regions.US_WEST_2);
        .build();

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

Amazon SQS Contoh Menggunakan AWS SDK untuk Java

Bagian ini memberikan contoh pemrograman Amazon SQSmenggunakan AWS SDK untuk Java.

> **Note**
>
> Contohnya hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. Kode contoh lengkap tersedia di GitHub. Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

Bekerja dengan Amazon SQS Antrian Pesan

Antrian pesan adalah wadah logis yang digunakan untuk mengirim pesan dengan andal. Amazon SQS Ada dua jenis antrian: standar dan first-in, first-out (FIFO). Untuk mempelajari lebih lanjut tentang antrian dan perbedaan di antara jenis-jenis ini, lihat Panduan Amazon SQS Pengembang.

Topik ini menjelaskan cara membuat, membuat daftar, menghapus, dan mendapatkan URL Amazon SQS antrian dengan menggunakan. AWS SDK untuk Java

Buat Antrian

Gunakan `createQueue` metode klien AmazonSQS, menyediakan CreateQueueRequestobjek yang menjelaskan 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 yang disederhanakan`createQueue`, yang hanya membutuhkan nama antrian, untuk membuat antrian standar.

```
sqs.createQueue("MyQueue" + new Date().getTime());
```

Lihat contoh lengkapnya di GitHub.

Daftar Antrian

Untuk membuat daftar Amazon SQS antrian untuk akun Anda, hubungi metode klien AmazonSQS. `listQueues`

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` kelebihan beban tanpa parameter apa pun mengembalikan semua antrian. Anda dapat memfilter hasil yang dikembalikan dengan meneruskannya `ListQueuesRequest` objek.

Impor

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

Kode

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
String name_prefix = "Queue";
lq_result = sqs.listQueues(new ListQueuesRequest(name_prefix));
System.out.println("Queue URLs with prefix: " + name_prefix);
for (String url : lq_result.getQueueUrls()) {
    System.out.println(url);
}
```

Lihat contoh lengkapnya di GitHub.

Dapatkan URL untuk Antrian

Panggil metode klien AmazonSQS. `getQueueUrl`

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();
```

Lihat contoh lengkapnya di GitHub.

Hapus Antrian

Berikan URL antrian ke metode klien AmazonSQS. `deleteQueue`

Impor

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

Kode

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

```
sqs.deleteQueue(queue_url);
```

Lihat contoh lengkapnya di GitHub.

Info Selengkapnya

- Cara Kerja Amazon SQS Antrian di Panduan Pengembang Amazon SQS
- CreateQueuedi Referensi Amazon SQS API
- GetQueueUrldi Referensi Amazon SQS API
- ListQueuesdi Referensi Amazon SQS API
- DeleteQueuesdi Referensi Amazon SQS API

Mengirim, Menerima, dan Menghapus Pesan Amazon SQS

Topik ini menjelaskan cara mengirim, menerima, dan menghapus Amazon SQS pesan. Pesan selalu dikirimkan menggunakan SQS Queue.

Kirim Pesan

Tambahkan satu pesan ke Amazon SQS antrian dengan memanggil metode klien AmazonSQS. `sendMessage` Berikan SendMessageRequestobjek yang berisi URL antrian, isi pesan, dan nilai penundaan opsional (dalam hitungan detik).

Impor

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

Kode

```
SendMessageRequest send_msg_request = new SendMessageRequest()
        .withQueueUrl(queueUrl)
        .withMessageBody("hello world")
        .withDelaySeconds(5);
sqs.sendMessage(send_msg_request);
```

Lihat contoh lengkapnya di GitHub.

Kirim Beberapa Pesan Sekaligus

Anda dapat mengirim lebih dari satu pesan dalam satu permintaan. Untuk mengirim beberapa pesan, gunakan `sendMessageBatch` metode klien AmazonSQS, yang mengambil SendMessageBatchRequestberisi URL antrian dan daftar pesan (masing-masing a SendMessageBatchRequestEntry) untuk dikirim. Anda juga dapat menetapkan nilai penundaan opsional per pesan.

Impor

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
import com.amazonaws.services.sqs.model.SendMessageBatchRequestEntry;
```

Kode

```
SendMessageBatchRequest send_batch_request = new SendMessageBatchRequest()
        .withQueueUrl(queueUrl)
        .withEntries(
                new SendMessageBatchRequestEntry(
                        "msg_1", "Hello from message 1"),
                new SendMessageBatchRequestEntry(
                        "msg_2", "Hello from message 2")
                        .withDelaySeconds(10));
sqs.sendMessageBatch(send_batch_request);
```

Lihat contoh lengkapnya di GitHub.

Menerima Pesan

Ambil pesan apa pun yang saat ini berada dalam antrian dengan memanggil `receiveMessage` metode klien AmazonSQS, meneruskannya URL antrian. Pesan dikembalikan sebagai daftar objek Pesan.

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 Diterima

Setelah menerima pesan dan memproses isinya, hapus pesan dari antrian dengan mengirimkan alamat tanda terima pesan dan URL antrian ke metode klien AmazonSQS. `deleteMessage`

Kode

```
for (Message m : messages) {
    sqs.deleteMessage(queueUrl, m.getReceiptHandle());
}
```

Lihat contoh lengkapnya di GitHub.

Info Selengkapnya

- Cara Kerja Amazon SQS Antrian di Panduan Pengembang Amazon SQS
- SendMessagedi Referensi Amazon SQS API
- SendMessageBatchdi Referensi Amazon SQS API
- ReceiveMessagedi Referensi Amazon SQS API
- DeleteMessagedi Referensi Amazon SQS API

Mengaktifkan Polling Panjang untuk Antrian Pesan Amazon SQS

Amazon SQS menggunakan polling singkat secara default, hanya menanyakan sebagian dari server —berdasarkan distribusi acak berbobot—untuk menentukan apakah ada pesan yang tersedia untuk dimasukkan dalam respons.

Polling panjang membantu mengurangi biaya penggunaan Anda Amazon SQS dengan mengurangi jumlah tanggapan kosong ketika tidak ada pesan yang tersedia untuk dikembalikan sebagai balasan ReceiveMessage atas permintaan yang dikirim ke Amazon SQS antrian dan menghilangkan tanggapan kosong palsu.

> **Note**
>
> Anda dapat mengatur frekuensi polling yang panjang dari 1-20 detik.

Mengaktifkan Polling Panjang saat Membuat Antrian

Untuk mengaktifkan polling panjang saat membuat Amazon SQS antrian, setel `ReceiveMessageWaitTimeSeconds` atribut pada CreateQueueRequestobjek sebelum memanggil metode kelas AmazonSQS. `createQueue`

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
```

Kode

```
final AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();

// Enable long polling when creating a queue
CreateQueueRequest create_request = new CreateQueueRequest()
        .withQueueName(queue_name)
        .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");

try {
    sqs.createQueue(create_request);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Lihat contoh lengkapnya di GitHub.

Mengaktifkan Polling Panjang pada Antrian yang Ada

Selain mengaktifkan polling panjang saat membuat antrian, Anda juga dapat mengaktifkannya pada antrian yang ada dengan menyetel SetQueueAttributesRequestsebelum memanggil metode kelas `ReceiveMessageWaitTimeSeconds` AmazonSQS. `setQueueAttributes`

Impor

```
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Kode

```
SetQueueAttributesRequest set_attrs_request = new SetQueueAttributesRequest()
        .withQueueUrl(queue_url)
        .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");
sqs.setQueueAttributes(set_attrs_request);
```

Lihat contoh lengkapnya di GitHub.

Mengaktifkan Polling Panjang pada Tanda Terima Pesan

Anda dapat mengaktifkan polling panjang saat menerima pesan dengan mengatur waktu tunggu dalam hitungan detik pada ReceiveMessageRequestyang Anda berikan ke metode kelas AmazonSQS. `receiveMessage`

> **Note**
>
> Anda harus memastikan bahwa batas waktu permintaan AWS klien lebih besar dari waktu polling maksimum yang panjang (20-an) sehingga `receiveMessage` permintaan Anda tidak habis saat menunggu acara jajak pendapat berikutnya!

Impor

```
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
```

Kode

```
ReceiveMessageRequest receive_request = new ReceiveMessageRequest()
        .withQueueUrl(queue_url)
        .withWaitTimeSeconds(20);
sqs.receiveMessage(receive_request);
```

Lihat contoh lengkapnya di GitHub.

Info Selengkapnya

- Amazon SQS Polling Panjang di Panduan Amazon SQS Pengembang
- CreateQueuedi Referensi Amazon SQS API

- ReceiveMessagedi Referensi Amazon SQS API
- SetQueueAttributesdi Referensi Amazon SQS API

Mengatur Batas Waktu Visibilitas di Amazon SQS

Ketika pesan diterima Amazon SQS, pesan tetap berada di antrian sampai dihapus untuk memastikan penerimaan. Pesan yang diterima, tetapi tidak dihapus, akan tersedia dalam permintaan berikutnya setelah batas waktu visibilitas tertentu untuk membantu mencegah pesan diterima lebih dari satu kali sebelum dapat diproses dan dihapus.

> **Note**
>
> Saat menggunakan antrian standar, batas waktu 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 sekali.

Menyetel Batas Waktu Visibilitas Pesan untuk Satu Pesan

Ketika Anda telah menerima pesan, Anda dapat mengubah batas waktu visibilitasnya dengan meneruskan pegangan tanda terima dalam ChangeMessageVisibilityRequestmetode kelas AmazonSQS. `changeMessageVisibility`

Impor

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

Kode

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

// Get the receipt handle for the first message in the queue.
String receipt = sqs.receiveMessage(queue_url)
                    .getMessages()
                    .get(0)
                    .getReceiptHandle();
```

```
sqs.changeMessageVisibility(queue_url, receipt, timeout);
```

Lihat contoh lengkapnya di GitHub.

Menyetel Batas Waktu Visibilitas Pesan untuk Beberapa Pesan Sekaligus

Untuk mengatur batas waktu visibilitas pesan untuk beberapa pesan sekaligus, buat daftar ChangeMessageVisibilityBatchRequestEntryobjek, masing-masing berisi string ID unik dan pegangan tanda terima. Kemudian, berikan daftar ke `changeMessageVisibilityBatch` metode kelas Amazon SQS klien.

Impor

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ChangeMessageVisibilityBatchRequestEntry;
import java.util.ArrayList;
import java.util.List;
```

Kode

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

List<ChangeMessageVisibilityBatchRequestEntry> entries =
    new ArrayList<ChangeMessageVisibilityBatchRequestEntry>();

entries.add(new ChangeMessageVisibilityBatchRequestEntry(
            "unique_id_msg1",
            sqs.receiveMessage(queue_url)
               .getMessages()
               .get(0)
               .getReceiptHandle())
        .withVisibilityTimeout(timeout));

entries.add(new ChangeMessageVisibilityBatchRequestEntry(
            "unique_id_msg2",
            sqs.receiveMessage(queue_url)
               .getMessages()
               .get(0)
               .getReceiptHandle())
        .withVisibilityTimeout(timeout + 200));
```

```
sqs.changeMessageVisibilityBatch(queue_url, entries);
```

Lihat contoh lengkapnya di GitHub.

Info Selengkapnya

- Batas Waktu Visibilitas di Panduan Pengembang Amazon SQS
- SetQueueAttributesdi Referensi Amazon SQS API
- GetQueueAttributesdi Referensi Amazon SQS API
- ReceiveMessagedi Referensi Amazon SQS API
- ChangeMessageVisibilitydi Referensi Amazon SQS API
- ChangeMessageVisibilityBatchdi Referensi Amazon SQS API

Menggunakan Antrian Surat Mati di Amazon SQS

Amazon SQS memberikan dukungan untuk antrian surat mati. Antrian surat mati adalah antrian yang dapat ditargetkan antrian (sumber) lain untuk pesan yang tidak dapat diproses dengan sukses. 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 huruf mati harus jenis antrian yang sama (FIFO atau standar) dengan antrian sumber.
- Antrian huruf mati harus dibuat menggunakan yang sama Akun AWS dan wilayah sebagai antrian sumber.

Di sini kita membuat dua Amazon SQS antrian identik, salah satunya akan berfungsi sebagai antrian surat mati:

Impor

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

```
import com.amazonaws.services.sqs.model.AmazonSQSException;
```

Kode

```
final AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();

// Create source queue
try {
    sqs.createQueue(src_queue_name);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}

// Create dead-letter queue
try {
    sqs.createQueue(dl_queue_name);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Lihat contoh lengkapnya di GitHub.

Menunjuk Antrian Surat Mati untuk Antrian Sumber

Untuk menetapkan antrian huruf mati, Anda harus terlebih dahulu membuat kebijakan redrive, lalu menyetel kebijakan dalam atribut antrian. 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 menyetel kebijakan redrive untuk antrian sumber Anda, panggil `setQueueAttributes` metode kelas AmazonSQS dengan SetQueueAttributesRequestobjek yang Anda tetapkan `RedrivePolicy` atributnya dengan kebijakan redrive JSON Anda.

Impor

```
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Kode

```
String dl_queue_url = sqs.getQueueUrl(dl_queue_name)
                         .getQueueUrl();

GetQueueAttributesResult queue_attrs = sqs.getQueueAttributes(
        new GetQueueAttributesRequest(dl_queue_url)
            .withAttributeNames("QueueArn"));

String dl_queue_arn = queue_attrs.getAttributes().get("QueueArn");

// Set dead letter queue with redrive policy on source queue.
String src_queue_url = sqs.getQueueUrl(src_queue_name)
                          .getQueueUrl();

SetQueueAttributesRequest request = new SetQueueAttributesRequest()
        .withQueueUrl(src_queue_url)
        .addAttributesEntry("RedrivePolicy",
                "{\"maxReceiveCount\":\"5\", \"deadLetterTargetArn\":\""
                + dl_queue_arn + "\"}");

sqs.setQueueAttributes(request);
```

Lihat contoh lengkapnya di GitHub.

Info Selengkapnya

- Menggunakan Antrian Amazon SQS Dead Letter di Panduan Pengembang Amazon SQS
- SetQueueAttributesdi Referensi Amazon SQS API

Amazon SWF Contoh Menggunakan AWS SDK untuk Java

Amazon SWFadalah layanan manajemen alur kerja yang membantu pengembang membangun dan menskalakan alur kerja terdistribusi yang dapat memiliki langkah paralel atau sekuensial yang terdiri dari aktivitas, alur kerja anak, atau bahkan tugas Lambda.

Ada dua cara untuk bekerja dengan Amazon SWF menggunakan AWS SDK untuk Java, dengan menggunakan objek klien SWF, atau dengan menggunakan AWS Flow Framework untuk Java. AWS Flow Framework Untuk Java lebih sulit untuk dikonfigurasi pada awalnya, karena menggunakan anotasi yang berat dan bergantung pada pustaka tambahan seperti aspectJ dan Spring Framework.

Namun, untuk proyek besar atau kompleks, Anda akan menghemat waktu pengkodean dengan menggunakan AWS Flow Framework for Java. Untuk informasi selengkapnya, lihat Panduan Pengembang AWS Flow Framework untuk Java.

Bagian ini memberikan contoh pemrograman Amazon SWF dengan menggunakan AWS SDK untuk Java klien secara langsung.

Topik

Dasar-dasar SWF

Ini adalah pola umum untuk bekerja dengan Amazon SWF menggunakan AWS SDK untuk Java. Ini dimaksudkan terutama untuk referensi. Untuk tutorial pengantar yang lebih lengkap, lihat Membangun Amazon SWF Aplikasi Sederhana.

Dependensi

Amazon SWF Aplikasi dasar akan memerlukan dependensi berikut, yang disertakan dengan: AWS SDK untuk Java

- aws-java-sdk-1.12.*.jar
- commons-logging-1.2.*.jar
- httpclient-4.3.*.jar
- httpcore-4.3.*.jar
- jackson-anotasi-2.12.*.jar
- jackson-core-2.12.*.jar
- jackson-databind-2.12.*.jar
- joda-time-2.8.*.jar

> **Note**
>
> Nomor versi paket-paket ini akan berbeda tergantung pada versi SDK yang Anda miliki, tetapi versi yang disertakan dengan SDK telah diuji kompatibilitasnya, dan merupakan versi yang harus Anda gunakan.

AWS Flow Framework untuk aplikasi Java memerlukan pengaturan tambahan, dan dependensi tambahan. Lihat Panduan Pengembang Java AWS Flow Framework untuk informasi selengkapnya tentang penggunaan kerangka kerja.

Impor

Secara umum, Anda dapat menggunakan impor berikut untuk pengembangan kode:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Namun, ini adalah praktik yang baik untuk mengimpor hanya kelas yang Anda butuhkan. Anda mungkin akan berakhir menentukan kelas tertentu di ruang `com.amazonaws.services.simpleworkflow.model` kerja:

```
import com.amazonaws.services.simpleworkflow.model.PollForActivityTaskRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskCompletedRequest;
import com.amazonaws.services.simpleworkflow.model.RespondActivityTaskFailedRequest;
import com.amazonaws.services.simpleworkflow.model.TaskList;
```

Jika Anda menggunakan AWS Flow Framework untuk Java, Anda akan mengimpor kelas dari `com.amazonaws.services.simpleworkflow.flow` ruang kerja. Sebagai contoh:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.flow.ActivityWorker;
```

> **Note**
>
> The AWS Flow Framework for Java memiliki persyaratan tambahan di luar basis AWS SDK untuk Java. Untuk informasi selengkapnya, lihat Panduan Pengembang AWS Flow Framework untuk Java.

Menggunakan kelas klien SWF

Antarmuka dasar Anda Amazon SWF adalah melalui AmazonSimpleWorkflowAsyncClientkelas AmazonSimpleWorkflowClientatau. Perbedaan utama antara ini adalah bahwa `\*AsyncClient` kelas mengembalikan objek Future untuk pemrograman bersamaan (asinkron).

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

Membangun Amazon SWF Aplikasi Sederhana

Topik ini akan memperkenalkan Anda pada Amazon SWFaplikasi pemrograman dengan AWS SDK untuk Java, sambil menyajikan beberapa konsep penting di sepanjang jalan.

Tentang contoh

Proyek contoh akan membuat alur kerja dengan satu aktivitas yang menerima data alur kerja yang melewati AWS cloud (Dalam tradisi HelloWorld, itu akan menjadi nama seseorang untuk disapa) dan kemudian mencetak salam sebagai tanggapan.

Meskipun ini tampaknya sangat sederhana di permukaan, Amazon SWF aplikasi terdiri dari sejumlah bagian yang bekerja bersama:

- Domain, digunakan sebagai wadah logis untuk data eksekusi alur kerja Anda.
- Satu atau beberapa 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 decider, yang melakukan polling untuk tugas keputusan dan menjadwalkan kegiatan atau alur kerja anak sebagai tanggapan.
- Satu atau lebih kegiatan, yang masing-masing mewakili unit kerja dalam alur kerja.
- Pekerja aktivitas yang melakukan polling untuk tugas aktivitas dan menjalankan metode aktivitas sebagai respons.
- Satu atau beberapa daftar tugas, yang merupakan antrian yang dikelola oleh Amazon SWF digunakan untuk mengeluarkan permintaan ke alur kerja dan pekerja aktivitas. 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:

- The AWS SDK untuk 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 diikuti. Informasi lebih lanjut tentang mengkonfigurasi dan menggunakan AWS SDK untuk Java dengan berbagai sistem build disediakan di Memulai.
> Demikian juga, tetapi dengan lebih banyak usaha, langkah-langkah yang ditampilkan di sini dapat diimplementasikan menggunakan salah satu AWS SDKs dengan dukungan untuk Amazon SWF.

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

AWS Akses

Agar berhasil mengerjakan tutorial ini, Anda harus memiliki akses ke portal AWS akses seperti yang dijelaskan di bagian pengaturan 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 kredensyal sementara, `credentials` file Anda akan terlihat mirip 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 isinya, 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 untuknya di dalam `<dependencies>` 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 ini ke proyek Anda (baik sebelum atau sesudah `<dependencies>` pemblokiran) 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 tipe domain, aktivitas, dan alur kerja proyek, dibagikan dengan komponen lainnya. Ini juga menangani pendaftaran jenis ini dengan SWF.
- ActivityWorker.java --berisi pekerja aktivitas, yang melakukan polling untuk tugas aktivitas dan menjalankan aktivitas sebagai respons.
- WorkflowWorker.java --berisi workflow worker (decider), yang melakukan polling untuk tugas keputusan dan menjadwalkan aktivitas baru.
- WorkflowStarter.java --berisi 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 menampung kelas Java Anda akan memiliki beberapa kesamaan. Demi kepentingan waktu, langkah-langkah ini akan tersirat setiap kali Anda menambahkan file baru ke proyek:

1. Buat file di `src/main/java/aws/example/helloswf/` direktori proyek.
2. Tambahkan `package` deklarasi ke awal setiap file untuk mendeklarasikan namespace-nya. 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.*;
```

Daftarkan 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 aktivitas dan jenis alur kerja Anda, 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, buat instance dari AmazonSimpleWorkflowClientkelas. Ini adalah antarmuka dasar untuk Amazon SWF metode yang disediakan oleh AWS SDK untuk 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 mereka 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!");
    }
```

Saat Anda mendaftarkan domain, Anda memberinya nama (kumpulan 1 - 256 karakter tidak termasuk`:`,, `/`|, karakter kontrol atau string literal '`arn') dan periode retensi, yang merupakan jumlah hari yang Amazon SWF akan menyimpan data riwayat eksekusi alur kerja Anda setelah eksekusi alur kerja selesai. Periode retensi eksekusi alur kerja maksimum adalah 90 hari. Untuk informasi selengkapnya, lihat RegisterDomainRequest.

Jika domain dengan nama itu sudah ada, a DomainAlreadyExistsExceptiondinaikkan. Karena kami tidak peduli jika domain telah dibuat, kami dapat mengabaikan pengecualian.

> **Note**
>
> Kode ini menunjukkan pola umum saat bekerja dengan AWS SDK untuk Java metode, data untuk metode disediakan oleh kelas di `simpleworkflow.model` namespace, yang Anda buat instance dan isi 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 dengan nama dan versi, yang digunakan untuk mengidentifikasi aktivitas secara unik dari orang lain di domain tempat ia terdaftar. Aktivitas juga berisi sejumlah parameter opsional, seperti daftar tugas default yang digunakan untuk menerima tugas dan data dari SWF dan sejumlah batas waktu berbeda yang dapat Anda gunakan untuk menempatkan batasan pada berapa lama bagian yang berbeda dari eksekusi aktivitas dapat berlangsung. Untuk informasi selengkapnya, lihat RegisterActivityTypeRequest.

> **Note**
>
> Semua nilai batas waktu ditentukan dalam hitungan detik. Lihat Jenis Amazon SWF Timeout untuk deskripsi lengkap tentang bagaimana batas waktu memengaruhi eksekusi alur kerja Anda.

Jika jenis aktivitas yang Anda coba daftarkan sudah ada, TypeAlreadyExistsExceptionmaka akan muncul. Tambahkan fungsi untuk mendaftarkan jenis alur kerja baru. Alur kerja, juga dikenal sebagai decider 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. Untuk informasi selengkapnya, lihat RegisterWorkflowTypeRequest.

+

Jika jenis alur kerja yang Anda coba daftarkan sudah ada, maka akan muncul. TypeAlreadyExistsException Terakhir, buat kelas dapat dieksekusi dengan menyediakannya `main` metode, yang akan mendaftarkan domain, jenis aktivitas, dan jenis alur kerja secara bergantian:

+

```
    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 terdaftar, Anda tidak perlu menjalankannya lagi—jenis ini tetap ada hingga Anda tidak menggunakannya sendiri.

Melaksanakan pekerja aktivitas

Aktivitas adalah unit dasar kerja dalam alur kerja. Alur kerja menyediakan logika, penjadwalan aktivitas yang akan dijalankan (atau tindakan lain yang harus diambil) sebagai respons terhadap tugas keputusan. Alur kerja tipikal biasanya terdiri dari sejumlah aktivitas yang dapat berjalan secara sinkron, asinkron, atau kombinasi keduanya.

Pekerja aktivitas adalah sedikit kode yang melakukan polling untuk tugas aktivitas yang dihasilkan oleh 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 pekerja aktivitas 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 menahan klien SWF yang akan kita gunakan untuk berinteraksi dengan Amazon SWF:

   ```
       private static final AmazonSimpleWorkflow swf =

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

3. Tambahkan metode yang akan kita gunakan sebagai aktivitas:

   ```
   private static String sayHello(String input) throws Throwable {
       return "Hello, " + input + "!";
   ```

```
}
```

Aktivitas hanya mengambil string, menggabungkannya menjadi salam dan mengembalikan hasilnya. Meskipun ada sedikit kemungkinan bahwa aktivitas ini akan menimbulkan pengecualian, ada baiknya untuk merancang aktivitas yang dapat menimbulkan kesalahan jika terjadi kesalahan.

4. Tambahkan `main` metode yang akan kita gunakan sebagai metode polling tugas aktivitas. Kita akan memulainya dengan menambahkan beberapa kode untuk polling 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 yang akan digunakan dalam passed-in. PollForActivityTaskRequest

Setelah tugas diterima, kami mengambil pengenal unik untuk itu dengan memanggil metode tugas. `getTaskToken`

5. Selanjutnya, tulis beberapa kode untuk memproses tugas yang masuk. Tambahkan berikut ini ke `main` metode Anda, tepat setelah kode yang melakukan 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 dengan 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 kami merespons dengan memanggil `respondActivityTaskFailed` metode dengan RespondActivityTaskFailedRequestobjek, meneruskannya token tugas dan informasi tentang kesalahan.

> **Note**
>
> Kegiatan ini tidak akan ditutup dengan anggun jika dibunuh. Meskipun berada di luar cakupan tutorial ini, implementasi alternatif dari pekerja aktivitas ini disediakan dalam topik yang menyertainya, Mematikan Aktivitas dan Pekerja Alur Kerja dengan Anggun.

Menerapkan pekerja alur kerja

Logika alur kerja Anda berada dalam sepotong kode yang dikenal sebagai pekerja alur kerja. Pekerja alur kerja melakukan polling untuk tugas keputusan yang dikirim oleh Amazon SWF dalam domain, dan pada daftar tugas default, tempat jenis alur kerja terdaftar.

Ketika pekerja alur kerja menerima tugas, itu membuat semacam keputusan (biasanya apakah akan menjadwalkan aktivitas baru atau tidak) dan mengambil tindakan yang tepat (seperti penjadwalan 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. Deklarasikan `WorkflowWorker` kelas, dan buat 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 keputusan menggunakan metode klien `pollForDecisionTask` SWF. PollForDecisionTaskRequestIni memberikan detailnya.

```
    PollForDecisionTaskRequest task_request =
        new PollForDecisionTaskRequest()
            .withDomain(HelloTypes.DOMAIN)
            .withTaskList(new TaskList().withName(HelloTypes.TASKLIST));

    while (true) {
        System.out.println(
                "Polling for a decision task from the tasklist '" +
```

```
                HelloTypes.TASKLIST + "' in the domain '" +
                HelloTypes.DOMAIN + "'.");

        DecisionTask task = swf.pollForDecisionTask(task_request);

        String taskToken = task.getTaskToken();
        if (taskToken != null) {
            try {
                executeDecisionTask(taskToken, task.getEvents());
            } catch (Throwable th) {
                th.printStackTrace();
            }
        }
    }
```

Setelah tugas diterima, kita memanggil `getTaskToken` metodenya, yang mengembalikan string yang dapat digunakan untuk mengidentifikasi tugas. Jika token yang dikembalikan tidak`null`, maka kami memprosesnya lebih lanjut dalam `executeDecisionTask` metode, meneruskannya token tugas dan daftar HistoryEventobjek yang dikirim dengan tugas.

5. Tambahkan `executeDecisionTask` metode, ambil 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 pemrosesan tugas.
- String untuk menahan input alur kerja yang disediakan oleh acara WorkflowExecutionStarted ""
- hitungan aktivitas yang dijadwalkan dan terbuka (berjalan) untuk menghindari penjadwalan aktivitas yang sama ketika telah 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 peristiwa 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

memberikan input awal yang diberikan ke alur kerja. Dalam kasus ini, ini adalah bagian nama dari salam kami, sehingga disimpan dalam String untuk digunakan saat menjadwalkan aktivitas yang akan dijalankan.

- acara "ActivityTaskCompleted", yang dikirim setelah aktivitas yang dijadwalkan selesai. Data acara juga mencakup nilai pengembalian dari aktivitas yang telah diselesaikan. 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 membutuhkannya. Lihat deskripsi HistoryEventkelas untuk informasi tentang setiap jenis acara.

+ CATATAN: String dalam `switch` pernyataan diperkenalkan di Jawa 7. Jika Anda menggunakan versi Java yang lebih lama, Anda dapat menggunakan EventTypekelas untuk mengonversi yang `String` dikembalikan oleh `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 tersebut.
- 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 hal 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 klien SWF mengambil token tugas yang mengidentifikasi tugas serta daftar objek. `Decision`

Menerapkan alur kerja starter

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 input. Di sini, selain menentukan jenis domain dan alur kerja yang akan dijalankan, kami menyediakannya dengan:

- nama eksekusi alur kerja yang dapat dibaca manusia
- data input alur kerja (disediakan pada baris perintah dalam contoh kita)
- nilai batas waktu yang mewakili berapa lama, dalam detik, seluruh alur kerja harus dijalankan.

Objek Run yang `startWorkflowExecution` mengembalikan memberikan ID run, nilai yang dapat digunakan untuk mengidentifikasi eksekusi alur kerja tertentu ini dalam Amazon SWF riwayat eksekusi alur kerja Anda.

+ CATATAN: ID run dihasilkan oleh Amazon SWF, dan tidak sama dengan nama eksekusi alur kerja yang Anda berikan saat memulai eksekusi alur kerja.

Bangun contoh

Untuk membangun proyek contoh dengan Maven, buka `helloswf` direktori dan ketik:

```
mvn package
```

Hasilnya `helloswf-1.0.jar` akan dihasilkan di `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 instance 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 pustaka 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 menjalankan contoh dan mendapatkan serangkaian kesalahan "NoClassDefFound", kemungkinan karena classpath disetel dengan tidak benar.

Daftarkan 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 di Daftarkan 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.'
```

Mulai aktivitas dan alur kerja pekerja

Sekarang tipe telah terdaftar, Anda dapat memulai aktivitas dan alur kerja pekerja. Ini akan terus berjalan dan melakukan polling untuk tugas sampai mati, jadi Anda harus menjalankannya di jendela terminal terpisah, atau, jika Anda menjalankan di Linux, macOS, atau Unix, Anda dapat menggunakan & operator untuk menyebabkan masing-masing dari mereka menelurkan proses terpisah saat 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.

Mulai 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. Sebagai contoh:
>
> ```
> echo "## Running $className..."
> ```

Setelah Anda memulai eksekusi alur kerja, Anda harus mulai melihat output yang dikirimkan 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

- Pekerja yang disajikan di sini dapat mengakibatkan tugas yang hilang jika mereka dimatikan saat jajak pendapat alur kerja masih berlangsung. Untuk mengetahui cara menutup pekerja dengan anggun, lihat Mematikan Aktivitas dan Alur Kerja Pekerja dengan Anggun.
- Untuk mempelajari selengkapnya Amazon 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 Java AWS Flow Framework untuk.

Lambda Tugas

Sebagai alternatif, atau bersama dengan, Amazon SWF aktivitas, Anda dapat menggunakan fungsi Lambda untuk mewakili unit kerja dalam alur kerja Anda, dan menjadwalkannya mirip dengan aktivitas.

Topik ini berfokus pada bagaimana menerapkan Amazon SWF Lambda tugas menggunakan AWS SDK untuk Java. Untuk informasi selengkapnya tentang Lambda tugas secara umum, lihat AWS Lambda Tugas di Panduan Amazon SWF Pengembang.

Siapkan peran IAM lintas layanan untuk menjalankan fungsi Lambda Anda

Sebelum Amazon SWF dapat menjalankan Lambda fungsi Anda, Anda perlu mengatur peran IAM untuk memberikan Amazon SWF izin untuk menjalankan Lambda fungsi atas nama Anda. Untuk informasi selengkapnya tentang cara melakukannya, lihat AWS Lambda Tugas.

Anda akan memerlukan Nama Sumber Daya Amazon (ARN) dari peran IAM ini saat Anda mendaftarkan alur kerja yang akan menggunakan tugas. Lambda

Buat Lambda fungsi

Anda dapat menulis Lambda fungsi dalam sejumlah bahasa yang berbeda, termasuk Java. Untuk informasi selengkapnya tentang cara membuat, menyebarkan, dan menggunakan Lambda fungsi, lihat Panduan AWS Lambda Pengembang.

> **Note**
>
> Tidak masalah bahasa apa yang Anda gunakan untuk menulis Lambda fungsi Anda, itu dapat dijadwalkan dan dijalankan oleh Amazon SWF alur kerja apa pun, terlepas dari bahasa tempat kode alur kerja Anda ditulis. Amazon SWF menangani rincian menjalankan fungsi dan meneruskan data ke dan dari itu.

Berikut adalah Lambda fungsi sederhana yang dapat digunakan sebagai pengganti aktivitas dalam Membangun Amazon SWF Aplikasi Sederhana.

- Versi ini ditulis JavaScript, yang dapat dimasukkan langsung menggunakan Konsol Manajemen AWS:

```
exports.handler = function(event, context) {
    context.succeed("Hello, " + event.who + "!");
};
```

- Berikut adalah fungsi yang sama yang ditulis di Java, yang juga dapat 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 selengkapnya tentang penerapan fungsi Java ke Lambda, lihat Membuat Paket Deployment (Java) di AWS Lambda Panduan Pengembang. Anda juga akan ingin melihat bagian berjudul Model Pemrograman untuk Lambda Fungsi Authoring di Java.

Lambda fungsi mengambil peristiwa atau objek input sebagai parameter pertama, dan objek konteks sebagai yang kedua, yang memberikan informasi tentang permintaan untuk menjalankan Lambda fungsi. Fungsi khusus ini mengharapkan input berada di JSON, dengan `who` bidang yang disetel ke nama yang digunakan untuk membuat salam.

Daftarkan alur kerja untuk digunakan dengan Lambda

Agar alur kerja menjadwalkan Lambda fungsi, Anda harus memberikan nama peran IAM yang memberikan Amazon SWF izin untuk memanggil Lambda fungsi. Anda dapat mengatur ini selama pendaftaran alur kerja dengan menggunakan `withDefaultLambdaRole` atau `setDefaultLambdaRole` metode. RegisterWorkflowTypeRequest

```
System.out.println("** Registering the workflow type '" + WORKFLOW + "-" +
 WORKFLOW_VERSION
        + "'.");
try {
    swf.registerWorkflowType(new RegisterWorkflowTypeRequest()
        .withDomain(DOMAIN)
        .withName(WORKFLOW)
        .withDefaultLambdaRole(lambda_role_arn)
        .withVersion(WORKFLOW_VERSION)
        .withDefaultChildPolicy(ChildPolicy.TERMINATE)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskStartToCloseTimeout("30"));
}
catch (TypeAlreadyExistsException e) {
```

Jadwalkan Lambda tugas

Menjadwalkan Lambda tugas mirip dengan menjadwalkan aktivitas. Anda memberikan Keputusan dengan `ScheduleLambdaFunction` DecisionTypedan dengan ScheduleLambdaFunctionDecisionAttributes.

```
running_functions == 0 && scheduled_functions == 0) {
AWSLambda lam = AWSLambdaClientBuilder.defaultClient();
GetFunctionConfigurationResult function_config =
    lam.getFunctionConfiguration(
            new GetFunctionConfigurationRequest()
                .withFunctionName("HelloFunction"));
String function_arn = function_config.getFunctionArn();
```

```
ScheduleLambdaFunctionDecisionAttributes attrs =
    new ScheduleLambdaFunctionDecisionAttributes()
        .withId("HelloFunction (Lambda task example)")
        .withName(function_arn)
        .withInput(workflow_input);

decisions.add(
```

Di`ScheduleLambdaFuntionDecisionAttributes`, Anda harus memberikan nama, yang merupakan ARN dari Lambda fungsi yang akan dipanggil, dan id, yang merupakan nama yang Amazon SWF akan digunakan untuk mengidentifikasi Lambda fungsi dalam log riwayat.

Anda juga dapat memberikan input opsional untuk Lambda fungsi tersebut dan menyetel start untuk menutup nilai batas waktu, yang merupakan jumlah detik Lambda fungsi yang diizinkan untuk dijalankan sebelum menghasilkan `LambdaFunctionTimedOut` acara.

> **Note**
>
> Kode ini menggunakan AWSLambdaKlien untuk mengambil ARN dari Lambda fungsi, mengingat nama fungsi. Anda dapat menggunakan teknik ini untuk menghindari hard-coding ARN lengkap (yang menyertakan Akun AWS ID Anda) dalam kode Anda.

Tangani acara fungsi Lambda di decider Anda

Lambda tugas akan menghasilkan sejumlah peristiwa yang dapat Anda ambil tindakan saat melakukan polling untuk tugas keputusan di pekerja alur kerja Anda, sesuai dengan siklus hidup Lambda tugas Anda, dengan EventTypenilai-nilai seperti`LambdaFunctionScheduled`,, dan. `LambdaFunctionStarted` `LambdaFunctionCompleted` Jika Lambda fungsi gagal, atau membutuhkan waktu lebih lama untuk dijalankan daripada nilai batas waktu yang ditetapkan, Anda akan menerima salah satu `LambdaFunctionFailed` atau jenis `LambdaFunctionTimedOut` acara, masing-masing.

```
boolean function_completed = false;
String result = null;

System.out.println("Executing the decision task for the history events: [");
for (HistoryEvent event : events) {
    System.out.println("  " + event);
    EventType event_type = EventType.fromValue(event.getEventType());
```

```
    switch(event_type) {
    case WorkflowExecutionStarted:
        workflow_input =
            event.getWorkflowExecutionStartedEventAttributes()
                 .getInput();
        break;
    case LambdaFunctionScheduled:
        scheduled_functions++;
        break;
    case ScheduleLambdaFunctionFailed:
        scheduled_functions--;
        break;
    case LambdaFunctionStarted:
        scheduled_functions--;
        running_functions++;
        break;
    case LambdaFunctionCompleted:
        running_functions--;
        function_completed = true;
        result = event.getLambdaFunctionCompletedEventAttributes()
                      .getResult();
        break;
    case LambdaFunctionFailed:
        running_functions--;
        break;
    case LambdaFunctionTimedOut:
        running_functions--;
        break;
```

Menerima output dari Lambda fungsi Anda

Ketika Anda menerima `LambdaFunctionCompleted`EventType, you can retrieve your 0 function's return value by first calling `getLambdaFunctionCompletedEventAttributes on HistoryEventuntuk mendapatkan LambdaFunctionCompletedEventAttributesobjek, dan kemudian memanggil `getResult` metodenya untuk mengambil output dari Lambda fungsi:

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

Sumber lengkap untuk contoh ini

Anda dapat menelusuri sumber lengkap:github: `< awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf _lambda/> untuk contoh ini di Github di repositori. aws-java-developer-guide

Mematikan Aktivitas dan Alur Kerja Pekerja dengan Anggun

Topik Membangun Amazon SWF Aplikasi Sederhana menyediakan implementasi lengkap dari aplikasi alur kerja sederhana yang terdiri dari aplikasi pendaftaran, pekerja aktivitas dan alur kerja, dan starter alur kerja.

Kelas pekerja dirancang untuk berjalan terus menerus, polling untuk tugas yang dikirim oleh Amazon SWF untuk menjalankan kegiatan atau mengembalikan keputusan. Setelah permintaan jajak pendapat dibuat, Amazon SWF catat poller dan akan mencoba untuk menetapkan tugas untuk itu.

Jika pekerja alur kerja dihentikan selama polling panjang, Amazon SWF mungkin masih mencoba mengirim tugas ke pekerja yang dihentikan, sehingga tugas hilang (sampai tugas habis).

Salah satu cara untuk menangani situasi ini adalah dengan menunggu semua permintaan polling yang panjang untuk kembali sebelum pekerja dihentikan.

Dalam topik ini, kita akan menulis ulang activity worker dari`helloswf`, menggunakan shutdown hook Java untuk mencoba mematikan pekerja aktivitas secara anggun.

Berikut 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 polling yang ada dalam `main` fungsi dalam versi aslinya telah dipindahkan ke metodenya sendiri,`pollAndExecute`.

`main`Fungsi sekarang menggunakan CountDownLatchdalam hubungannya dengan kait shutdown untuk menyebabkan thread menunggu hingga 60 detik setelah penghentiannya diminta sebelum membiarkan thread dimatikan.

Mendaftarkan Domain

Setiap alur kerja dan aktivitas Amazon SWFmembutuhkan domain untuk dijalankan.

1. Buat RegisterDomainRequestobjek baru, berikan setidaknya nama domain dan periode retensi eksekusi alur kerja (parameter ini keduanya diperlukan).
2. Panggil AmazonSimpleWorkflowClientmetode.registerDomain dengan objek. RegisterDomainRequest
3. Tangkap DomainAlreadyExistsExceptionjika domain yang Anda minta sudah ada (dalam hal ini, biasanya tidak ada tindakan yang 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 membuat daftar Amazon SWFdomain yang terkait dengan akun dan AWS wilayah Anda berdasarkan jenis pendaftaran.

1. Buat ListDomainsRequestobjek, dan tentukan status pendaftaran domain yang Anda minati—ini diperlukan.
2. Panggil AmazonSimpleWorkflowClient.listDomains dengan objek. ListDomainRequest Hasil disediakan dalam suatu DomainInfosobjek.
3. Panggil getDomainInfosobjek yang dikembalikan untuk mendapatkan daftar DomainInfoobjek.
4. Panggil getName pada setiap DomainInfoobjek 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());
    }
}
```

Sampel Kode disertakan dengan SDK

Ini AWS SDK untuk Java dikemas dengan sampel kode yang menunjukkan banyak fitur SDK dalam program yang dapat dibangun dan dapat dijalankan. Anda dapat mempelajari atau memodifikasi ini untuk menerapkan AWS solusi Anda sendiri menggunakan AWS SDK untuk Java.

Cara Mendapatkan Sampel

Sampel AWS SDK untuk Java kode disediakan di direktori sampel SDK. Jika Anda mengunduh dan menginstal SDK menggunakan informasi di Siapkan AWS SDK untuk Java, Anda sudah memiliki sampel di sistem Anda.

Anda juga dapat melihat sampel terbaru di AWS SDK untuk Java GitHub repositori, di direktori src/samples.

Membangun dan Menjalankan Sampel Menggunakan Command Line

Sampel termasuk skrip build Ant 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 sampel GitHub, klik tombol Raw di tampilan kode sumber saat melihat file README.html sampel. Dalam mode mentah, HTML akan dirender sebagaimana dimaksud di browser Anda.

Prasyarat

Sebelum menjalankan salah satu AWS SDK untuk Java sampel, Anda perlu mengatur AWS kredensil Anda di lingkungan atau dengan AWS CLI, seperti yang ditentukan dalam Mengatur AWS Kredensial dan Wilayah untuk Pengembangan. Sampel menggunakan rantai penyedia kredensi default bila memungkinkan. Jadi dengan menetapkan kredensil Anda dengan cara ini, Anda dapat menghindari praktik berisiko memasukkan AWS kredensil Anda ke dalam file dalam direktori kode sumber (di mana mereka mungkin secara tidak sengaja diperiksa dan dibagikan secara publik).

Menjalankan Sampel

1. Ubah ke direktori yang berisi kode sampel. Misalnya, jika Anda berada di direktori root unduhan AWS SDK dan ingin menjalankan `AwsConsoleApp` sampel, Anda akan mengetik:

   ```
   cd samples/AwsConsoleApp
   ```

2. Bangun dan jalankan sampel dengan Ant. Target build default melakukan kedua tindakan tersebut, sehingga Anda cukup 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 IDE Eclipse

Jika Anda menggunakan AWS Toolkit for Eclipse, Anda juga dapat memulai proyek baru di Eclipse berdasarkan AWS SDK untuk Java atau menambahkan SDK ke proyek Java yang ada.

Prasyarat

Setelah menginstal AWS Toolkit for Eclipse, kami sarankan untuk mengonfigurasi Toolkit dengan kredensil keamanan Anda. Anda dapat melakukan ini kapan saja dengan memilih Preferensi dari menu Window di Eclipse, dan kemudian memilih bagian AWS Toolkit.

Menjalankan Sampel

1. Buka Eclipse.
2. Buat proyek AWS Java baru. Di Eclipse, pada menu File, pilih New, dan kemudian klik Project. Wizard Proyek Baru terbuka.
3. Perluas AWS kategori, lalu pilih AWS Java Project.
4. Pilih Berikutnya. Halaman pengaturan proyek ditampilkan.
5. Masukkan nama di kotak Nama Proyek. Grup AWS SDK untuk Java Sampel menampilkan sampel yang tersedia di SDK, seperti yang dijelaskan sebelumnya.
6. Pilih sampel yang ingin Anda sertakan dalam proyek Anda dengan memilih setiap kotak centang.
7. Masukkan AWS kredensil Anda. Jika Anda sudah mengonfigurasi AWS Toolkit for Eclipse dengan kredensi Anda, ini akan diisi secara otomatis.
8. Pilih Selesai. Proyek ini dibuat dan ditambahkan ke Project Explorer.
9. Pilih `.java` file sampel yang ingin Anda jalankan. Misalnya, untuk Amazon S3 sampel, pilih`S3Sample.java`.
10. Pilih Jalankan dari menu Jalankan.
11. Klik kanan proyek di Project Explorer, arahkan ke Build Path, lalu pilih Add Libraries.
12. Pilih AWS Java SDK, pilih Berikutnya, dan kemudian ikuti petunjuk di layar yang tersisa.

Keamanan untuk AWS SDK untuk Java

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

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

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

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

Topik

Perlindungan data di AWS SDK untuk Java 1.x

Model tanggung jawab bersama berlaku untuk perlindungan data dalam AWS produk atau layanan ini. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur

global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali terhadap konten yang di-hosting pada infrastruktur ini. Konten ini meliputi konfigurasi keamanan dan tugas-tugas manajemen untuk berbagai layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat FAQ Privasi Data. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting blog GDPR di Blog AWS Keamanan.

Untuk tujuan perlindungan data, kami menyarankan Anda untuk melindungi Akun AWS 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 AWS sumber daya.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, dengan semua kontrol keamanan default dalam AWS layanan.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan. Amazon S3
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi lebih lanjut tentang titik akhir FIPS yang tersedia, lihat Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak memasukkan informasi identifikasi sensitif apapun, seperti nomor rekening pelanggan Anda, ke dalam kolom isian teks bebas seperti kolom Nama. Ini termasuk ketika Anda bekerja dengan AWS produk atau layanan ini atau AWS layanan lain menggunakan konsol, API AWS CLI, atau AWS SDKs. 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 untuk Java dukungan untuk TLS

Informasi berikut hanya berlaku untuk implementasi Java SSL (implementasi SSL default di AWS SDK untuk 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 JVMs, 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/pengembang/tls-1-3- - -1-9-5-ke-1-10-31/incompatibility-with-aws-sdkfor-java-versions

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 berbasis OpenJDK JVMs, Anda dapat menggunakan properti sistem. `jdk.tls.client.protocols`

```
java app.jar -Djdk.tls.client.protocols=PROTOCOLS
```

Konsultasikan dokumentasi JVM Anda untuk mengetahui nilai PROTOKOL yang didukung.

Identity and Access Management

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

Topik

Audiens

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 untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM AWS, lihat panduan pengguna yang Layanan AWS Anda gunakan.

Administrator IAM – Jika Anda adalah administrator IAM, Anda sebaiknya mempelajari detail tentang cara 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 sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensil dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensional. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori perusahaan Anda, penyedia identitas web, atau Directory Service yang mengakses Layanan AWS menggunakan kredensil dari sumber identitas. Identitas terfederasi mengambil peran yang memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami menyarankan AWS IAM Identity Center. Untuk informasi selengkapnya, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih 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. Contohnya termasuk 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. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

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

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnya ACLs, lihat Ringkasan daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — Tentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCPs) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

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

Bagaimana Layanan AWS bekerja dengan IAM

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

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

Memecahkan masalah AWS identitas dan akses

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

Topik

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

Saya tidak berwenang untuk melakukan tindakan di AWS

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

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `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 orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah AWS mendukung fitur-fitur ini, 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 Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Validasi Kepatuhan untuk AWS Produk atau Layanan ini

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

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

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

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

Ketahanan untuk AWS Produk atau Layanan ini

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

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

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

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

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

Keamanan Infrastruktur untuk AWS Produk atau Layanan ini

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

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

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

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan principal 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 untuk 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 untuk 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 SDKs, 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 untuk 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 untuk 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 untuk 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 untuk 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 dari `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 (CEKs) 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 untuk 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 untuk Java

Semua artefak Maven yang tersedia untuk umum ditandatangani menggunakan standar AWS SDK untuk 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	0x AC1 07B386692DADD
Jenis	RSA
Size	4096/4096
Dibuat	30/06/2016
Kedaluwarsa	2026-09-27
ID Pengguna	AWS SDKs dan Alat < aws-dr-tools @amazon .com>
Sidik jari kunci	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 6692 DADD AC1

Untuk menyalin kunci publik OpenPGP berikut untuk SDK for Java ke clipboard, pilih ikon “Salin” di sudut kanan atas.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Hostname:
Version: Hockeypuck 2.2

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
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJo12ZrBQkTQxnmAAoJEKwQezhmktrdi18P/A3De83MBx8bdcWJ
Fot71Vk1TyBQFErgrtcytSU0czEHx3tGbzgQLbMlyzjirOT03usxEkOeqTVK+RU+
5uFXNZYQLwMJlHJ6S8tnfLe/ExM5WQ2KPwIUPfZs1GDDRQB2dIKSc+qYrP1O1vf4
O4iPgfLHMW2bFh3zjjxcaHCJyqc7Cau33eZFBAsRni1jOUo7MeyX0hlXfW8pd48Q
wZllQVZ/6KmDiFWA0CZ+2svJ5cL0tgPoh1OQjoz0nHpNfuDILMrZ+e7tx2VTlkGH
UGeNSydnrK8v9ztFn34KtU/k7NEWoVSyEi+5ICZL18FBwPqTwdVWXwXrqZCKiIpr
8ZdJWDz2sJfgDFNCC6rKgCQ6FrmaD9G76dYWkQ4AbZqABlUzU3q36W1K0r3iOAb5
G4tdOt4yqXHTe1x+ZUNaeW7gaCmtXAxLw0OfeJrcq/44b/SQP+qJ8sSOv76Yg2oF
BsF5DWOVUFghbTyokHAoVROyhBR4dUUisY39AqLSL8+Lp9Pr3wNuGl9GLrMD5701
piUb88B3Gwe1EiKV1gaKrvZ3mECDUiSMVO0Z5iG8E4QDpNmVbJbV1uT821ubvtOv
2Ko10Fa0uwCYGssdRGqEXNy6jz/Er8LAC3+nmGINDJQzrF+loYoSSkI2Nu7lhMuL
7iWwUPF7OhDXoVSAn4X3x6q2rGK0wsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZTsFCRNDGLYACgkQ
rBB7OGaS2t0/Ow//YIv51vHtD+kwMmIvk3zpizDHY0zW2dOezAo+C/DsSyC7wDll
Dixw34EQ1yLXH5xLR8CH1zupl3JmmEp1ucdQggoefbidxDl8Fld7tJOD1y3GGnTD
0jAl2ZC+W65Oh+wS1mD1FlaKjMGGkvJf0dA7RtU2T8dv3vt8dsxg76FMFS3+fqlC
FNOAsNTn9zWR1SqBIfkMJK83aq6s/rcEV9VrAYgDgqex58fygB5EuTf842/IF7WZ
Q9gd6fupB0mMZP5YWd2uj/vsBTYakG+mgQwDxZuKPeEzAqnqqS7biSQOUO6Wozlq
Yy4fSczE9GkBAvg0pGmbko+zHvpnjvX/h1CUpC6odvFyOAhZp6zyhs0QWz9thfqV
lU8WlbgJ2atFDn5GUSxF/fe0Yzovlbbs6sbYXuvMG9RiEOuJ1mBbZR3aIdZ1U6Do
BHc/vjc5mWcV7JQSP7i4W/8W7X3UAuN9LdxB+IvF3Cwrgtlw2BWvA5Alco5Tnz8t
P/CIVmBjk+sLme8W4kfLK3IWEbwClOdNnErI/MHRm65A2Y5EMIhwjrOi07SU1Pxa
nPpg3OYJCdvjzdB8QE3/DBiMfOl4dISfKDVEWnfK8mZaYd/BeRm2gUAa9UrqSFCG
BlA7Lg+eLI3US0FvwWJ4j5bBJqgLu+y7crIkiUOPAQuLk3lO+5uYU/I3DuLCwZQE
EwEKAD4CGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AWIQT+uSCfLy8/RmSEHlWs
EHs4ZpLa3QUCZwAXCAUJEWvKgwAKCRCsEHs4ZpLa3ZdTEACMBLg2q9zk8ZH02nDz
Sg5zc8Wlqq8WdxU0Pj8qx4UOrrMca7wyiUvrgoxPW5lh1RVNUeMkDRfu9pSXcOVI
V9LvmYE/WnwKROubgGbsC4T7M/LqV0/AulXil4d7IXcO6l4toa8LTNWtD5bODgrN
gvay1AzCU8kq1Qw1cKZ2gAfvA3Ba7PWyLeUN4HTlGrXcw73G+0CofY1L8wqWxHCJ
29XqQzeTEc6MDEeIlNlVdUcy8Qr5uwkEsl34H9AxS5F1opJ4TqvXiDZsrSRRv57R
XYmRZDWeYT+9PZaMsHXza5qgej7BfATxhYfICsNaY6MK3x6b+nDSKkoZgO+i09zh
1YjpahhQe6G336v/3mRj0dKGCRQ6znQ9ghUaB5z9zfvgH5AOEkTe3l8MqM+j5A6P
VjSBBJAHKejxr7+wKJKIA6P+DqpsYAunzftwUzrLVqb+BZQ+DcTmVrE7OPcMYJD5
QglX/Le+WmWZHI154NXgpWWUOUgZUbUge4DKrT+zCJ9iecPLKTW7OcULyXO+rjb8
8BGrD5GPlHB3dOUXXTlMKCqg3qy1Bu2KnZTQiaEEdZgSIGQbrW0JTMmmXJkKjokd
JMA4vYeg5en51G9nRQjScPngx77IxvByNyFWTJdG1ENpJpsK9TtmENcpyUJtJZTJ
```

```
ZSOIRVPP5RzR5vInuXWq6VV0BMLBlAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMB
AAIeAQIXgBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEoiBQkPj/2dAAoJEKwQ
ezhmktrdx1YP/0vvym3jgX/pwnR7K1rafZMb1iKQBr0ISG8cdbaf4pqX5vuUZnyj
w9Cl/oONn7jJjnQxOIIzuBoxne2WN28ftM2w0nVXm85mAmz2fwQz/fdKDyonXcOh
pfD2iMqn7gESjhEgRE7wMDYMDuLdqHI7OKWGVfgrh7xEmKapLh45h7cnumo2VjL9
uDYY1aOBHz993T7oE41y43rhk+6kKbGFd2uuo7h5j1ZF8Lj6sYfcEzXOU1OhRlD0
nyBjDy9MYWu0YNouc70WgMceGx6hjvCAM/5fxP7SZFecZ7ePeB0GpvVA24hSNENE
0r3tUekuOf1I0FunMnMnbh7ZO9rPYqWvWDNIpU3S4CjFhY82L+IeKnmLy8N6ASRk
HsPiNCOHSK8C/0ynrd9xLhX8Jsk/TGiQYaleoHhWkNLlZsL86QHL8SKEqkqZCQf5
AEqghDP6NEGS7lnOenA7JjIrA9KLlT7fnNWZOwFi5X+o/CymE2ytEMS0Yf3nmY4U
n9x56Wgn6J2zqB5nqOXf6NxGdAIgOBm098YEnKCIFzk+yhoDlprVpHcnd2b5f6Oq
uh8KYOEbKgpMJ3zZuWSL5kwGF1nNoYiAkonMaz9H3pOQnOMVYCUeUTDRsiO/prrd
UhNlry4TAsBMpeXnFhdLVM3vFQZVpByadGOJNmnaN/Wavw2a00UGBFa4wsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJhMqGaBQkLnlUVAAoJEKwQ
ezhmktrd2sQP/3YHM+U+BbOy1nSEAfykZ71+uCM2hkHMLdxQYWB/rBWkmg/pbu+d
r4t45RsTASrNjRcZOntlPMQRIq973ymHfpmeS+noFwvTGH7zDv1BRBR9wPrd1XUz
iSuEUHGi/fqxUVXQ5mbonzfThX8tuXeuiQmeToqoB00FYlZm6xsNnEHcjVl66mC4
IPoJLWnZJs4rOCeoRf5XvDTgX6xt5/kLYRZf79qaWGFvaZpsc1CH+rQJUdVa/D4T
7pI7hX6zy0S91z4iuC5HZUiOTF+y5auEZHGTdTWNS1kvOvfcCTi0XK/GkGL82SZu
7X2VGnpCeUnFyViRGlk+KaDG1sVyDY+lcBPg6ilr45M6MQV0iHS5OF04QNXSKt5+
UnzJH7lldgNsR6ibRMyNV3k5v3fyUcSBvIYyLORTTBiVEjQDSbk1QNqbrQlX9CWz
+EJWn16BFTmMFvxBSWPm640GncHP5J3/0MbMw3Cm90x7k8UfNANIemcrJrSxIDwm
g9cVAg3a+D+wxjrVe8jGg0ejvECpm+0yswigj5x6Lqj09A4UgdjEauN+/pn0nhBo
Gv7DzMXtM/LoDtgp6wn93qZVN2TsuHnkEk4UyntB6eWJbBdXHWUr47exiWh0dvQN
tpwCWPT6I7ZTPtA5K/zx+q9m6797BLgAkTYc6gloQL3vs1Z1S3m/hZNawsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJgmrz2BQkLBnBxAAoJEKwQ
ezhmktrd36oP/2rB2EkwSOCKC4m0heWSfDWi6OBKoEbbDtFtc6/HwqBW8SPsiKlq
zV0e3qBY/LVju04+ktJEK+EGXLnC3iC36MegrQ8zt391kEx/Zv9LIuVOCX90QIAX
dL8MVUkkjRLCFFH8pTgRy1cJYWk1X4dYdXWYc29fCwNVartNdNBhsb2ht3VJeKDE
kUivBHmkjuISDPEnI1coY7Lj0ZtY5cHdRF2eZpB0RkTBpsIt18rCYyHkERZrhmvb
j3rOyPyvOa+1/dQS8/hv5pEmbKx8cy8RdJkmbUHYatPBsjHkJSWr7O7G9VFW4GoN
9CRAI4KkbDSEDjCL5dv2pq0Sew1MkLuWJGULAMgiIUlWcOs5SZZGFSksNQrtSFV9
Z/wGocecMGkGQNXQ06JV/Fry/TvyphBlmylEqL+NLqEcEjnlz90IVu+ZA+M09J96
UlHO7V5GvBgM+QK/q/dJeMHPWrNlo1gA6Nwl/HBdM0DqzdZ2jEPvsQSABvZrPMty
+BAqEar4wqY1AH4X5ccEjO7nJQoBQSDRSki1fkBsc1nx44N/m0kHdIa0Z/Y+Mw4v
WiZhREkOospG1I4lBa3CNTVAhSs9msGsYfkqvFJGHL7sZY8XSv82GBBvA0nUNrsJ
bLBwo2FaQG9eoatRAGkqp4b/OtNtBuGeiQoNwFGbfUZTAaStj5/zZj0swsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJe+9bwBQkJZ4plAAoJEKwQ
ezhmktrd+ScP/RoaUKriVVAgLHOGs+/mnfKtnfTlClzi5dsdI9/6H0vLpmSWK/Cl
2cT6gary45VMgAeVK+H1lQXafYj+FY++I5kYoe2GrSvIXhpjaFAJyNf/dKleTsqR
Tm371i8b3FDYs5kvy2CnTbmHB8MsOGxck8/YHd1x+g8WpO2IgF89yYCSF3CAdxC3
6bHbs6Z3C3lcM/3SoWF+Yie2P8XeBMPCGp/BcjQzUcHF6G06TwDDYhixucUi6vEY
EH5JtOwVVQ7bubT8OFeOoJwVxlzYz4UoqxjKDWymarTzu03AUIT0PXPece94bJAK
mSh68ItQe3H8tSPMubERWz2tEV3lVkChDGXcC7BYQmxHseolxz/qzCtJ0iX9BvZR
```

```
dniZNeNJ/Cu8M2pDp47zdNFXzf/Q/sQ9pQlws22G2g119rWDneBku9n1vTP80/er
SB+VLTBjDiArlCY5y9+BG8wbscExJySoQxkB9j/nlMzPY5rgk0SyxsNj9GbqH+hr
EjS3/uacNwSLxGcOT2E9Teot5pfTEO6fQVq+35QhfAlP8c8jze01W/+u+wXu1Ui9
azRSzYtCHanGyyet6U1mlBpAkqkZzH6t3CA5czc9i6FbzjvFVZnbRUZIRzfISYew
lF5WqgTn2iYVdxagPRvLF5kjd696brGW9d5HwirCVGaK04VsXWlAblB9wsF9BBMB
CgAnBQJXdYAFAhsDBQkHhh+ABQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEKwQ
ezhmktrdWigP/3QWl7aO81BUWyby4HEhN4SdAoWGY/FLqO4mCtuplcnMgRUCSiL9
l2BSCTMCtUcdSWtYwOgSChN2mMsdi1U2FNR5HvNunYR/pFdqjfQurf1ZmKVeG5/4
uuKaOxMw9e8pK5uYAfs+O7gr8gu/f6/Drp7NZk3/yVKpf4WCY9oX9TA1q9O/11nN
cwS45U/d7YP+N1YM9cBXa1DnDcdfm0BlykzouAF0qd1Lwi/tmLENvybD3+2c2WsE
rlFZGSa5ZafO0tTIWXh5k6wh5FdRRycrnSyRK3B9N9+yaXfMQ0XpOypa8dqQEnCi
IsngDCJPxtTrhMWKhBFRUMzK/WZTDboTQSQDK+YVRrE4K8MtoZSKwZLV2r9O3TpX
kpbKsPVYmexerfdMeZfjZMF1bC7BmEs7jciH6JjbqAoAPnHzN0481aeNarINSViX
PQWr2mp9qShei2/RavLtx2ZNrvmGW72ZKpF8E3WWUDpBJqFVeGNRvOm3aZj8o/Hl
ewtNjcT4ouJfqlfKiULv+g7ANEMDLQTFDTg5twRdvmZlB7oTBsavf+LwxPIXhH32
IR7TX7VeicMMxmZnmZK2ANT/QBi3laf+ojVHvB+f6D74eLNq0Zqjfi/3UFNYsYjg
E+YgCqEUBpHbl61nOHwGOSsQwfap2uKKlzukD/KxH5SPBC3DYGBI+KCbzsFNBFd1
gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+hV6XulGA
HAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go7xHIxgFj
C046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FKVYR/j9ue
nEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQlQ1Kou+3
dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjypUwgp0MT
o25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGwsMDyHNqyJ
eYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6QxaZje9YSZU
ijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KDOSn5CbmX
pAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVgroUVtprs
mHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs/Hd981Fd
VghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQABwsF8BBgB
CgAmAhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZm4FCRNDGegACgkQrBB7
OGaS2t3y5g/7BFXp/fdanzuQPToJTPen7AVwhLloKaiYhG3GjdXfMPLvu6UtaaGm
qynLolUNNooobptFqc1G9BKoAghQrta7CsDHtsQF2xyc3Mfu0gmpL/7X5a7sFIeJ
j08UjfweHx4DSG4LEZgNaAoWFjZltp4+8cqijkAHXt+r+1ayQG4VVHOWyXXqmSH4
9HqtbPcPyRzxdoVLeshZC9jmhHhhKqw/LwGyipWSOUKQDjWarBwdyhNmWCaLvxH1
ndMp4tq8DPGC3G4T9tYAbANrn7nKfZgHebMSzMw9kSp0L6QvwwTDjJyIWz85WyeH
WHeBysDaBOit3XDlehUew27y7N6a9hQSYjnXuwvre5mjDIOqJon/31R6ui2Z1y9P
a+bC11hbLXXh9tLCXRuoOt6thh9Cq5X1a76PPpEv30o3bpsb6l2hbrut1OKezwvK
l7txito/jfMiWfsZHA9O4SoM+8GnmVingHtZ805n1T4RddJvT/vaqplfI6zf7jmf
a69lALP420riFOQcwntNUM5tVmFUZsnFp2YRd4Ls7MiXVjtABahlSbb94l5WSVc0
jrOLDf94edvzk4R8i2Ob8CfVZNqEsTR6bHz8dT7Q+xQzEdjUujyyZY1UUl157Qeb
OsHjhCtuZYCI04X9hZ37nKnZXSxRlRDCnt5BEiyFu2WD1RscUe6PcVDCwXwEGAEK
ACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCaNdlPQUJE0MYuAAKCRCsEHs4
ZpLa3XCpD/42DrcveE+q2ulrAIYPDlUlHiwIMejqBDRm6zmr1KSAeb4E6/MFcP4s
rXSSscMlrqG6NVynjNCXjD2YzWii68EwoXLJkgoD3r2ifzkV62EX2MIEeNZAVwuy
KNxorzmy6bhuWltRYNK/hITs2AG5orOk9ADEJ8PixKymrWlhesPaWX6Yhp9/tWaC
```

```
RHOSRiLbRVaJ+7sqT88urLmkV9Hqx949Zxv4+cgBVUGL6WXKsfWhHjbDMNJnozWB
SZaIJznLApOM8z+1DNrqUYyfR8SkF4IOVmg6HDzoyuseJJ8JvMAlkvT6F9VBq/iE
yeDYdEEQxwHwozKrEx5Ybxl5mntbqwCXy6kHSx2+/3RZWpZQ8K29YP9QEk0KeGF8
9Vap3jjNrX4u3cuRNQpeblQc4uFn3Nzaj+cVV4YzcRw94NifecXpujSvk8XU2ytJ
/JgMBxPIBKglN4eEMet9b4FRB5XeBdPAm19/LXyb4IlIipGNXlgNz/HCuBzidzHT
QQdqfA9rZVx1hwFr7AJCVqWaXVsx1oEAhKqpTtsLMyj594DvnRuwKw5Vse+1eydW
MIHYdbxmJccsTGIt/hsOpc8zfm+QYk5752jshhOKEBy+Ey3QZI1WbO547NOb2Hwr
Pgt7fw2NCKMPElSu98zmneFPhqNHf7L5urBe5gADj81E8lm6t/oVxcLBfAQYAQoA
JgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJnABcLBQkRa8qFAAoJEKwQezhm
ktrde3MP/13CLWp99XvRROrzD/bWOfWjAenT2PE/tYdOY9YcTQFbnIUhaVUDWAo3
pibR3D4u9LlY4olpGfJ7BTIHFa9myfpaVvmrNjueYI4omli24JQ/CKqNdY8Qzxz+
/QyiNK7Aw5cEBWIu84WGB1SsefWWT3rZe9YBb77gNcWHZ15pXTXrcgUxGY4808MC
I9YFWq8EA0iHawtFnmB3UFfClWt37Hy3PKvr1is3uG60+ULI8RQz3/+ZwSG8U+xt
b+I7H9+gITc1eFCb+tIwp5xWflyxcFXYk6UzOL7y3Fg2tIEuSNtIHUC9NDVobf6c
I0KAzZcMvKiPQiuBnVOjgDLmCZM5H6axj9x+gi4oVh6ea3HLqMzyjm5JkeCGgKWv
H0gD3yGEZDvcbavkQOle5T+4JefndKzCPrluX0iyx+oQiiOL8WieSSkSB6BsZcUN
SeuGJwM79Y7Oqld/YVrQNBZj5Vz+m3nZ+0EWDDMI0hRgMpSEIc+dnTC0u103Z+Rc
c2IJq8INmU653sUcfCZE12ParW4rF7ib6kViYrABT8f4e2TP0aOyP5kp51ied9qL
azaBA6tt/C9X1V2EJZK4srXtmcZO2Im45RAiVXyfpBAmmiF3eZWCbKe7qBC4rDRh
LZG4RQW/S86Da0BID7gQz9IFSkaG504MsDhvnA7iAqaHUHUepCsiwsF8BBgBCgAm
AhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiQFCQ+P/Z8ACgkQrBB7OGaS
2t3AwA/9GkXKUgvjKGCxwE4SdDt7c2jw6to2TTP9iFJ3Xbk3+5BURT3gkZCuu9D7
gt+97aVo/B4EM7Xz8DQKyY7Ic9VAwDRra/Hwi1V0hw1zyIWQ/gAnX3baU6qLRWHR
vVR5meV8r35C+rg9DaWFYmvS7PIv9LfxESwBPUjbmx8k4/5EJpHUwf12bzkTnot5
7q5lHxKQa6IvqQak+Hp9ZM2KPdsgKO2HWJJIIvYcI5byW9zBKV0O7YR8gtRAJKp9
IbtsXx0WT6cqHOFVc5SSzdcaMt0gLFl7BTnJyvKK2l9GABGBmzYDjeCyF2J+Ippf
oqxqfTe6EoOsuEMc2PbLTs9SsWjyCC2VGlX8+uUH9SoKwL0VQ6LFsP6fhkVKqi/a
rB6UuPR/iZnrKIuxMNQ4U+t2Q6UdMlmXsAXTNdkwzoK9oJRokIrH0ZV1KtH4sjjA
tCic+tOddq+GQLiKe2WpJfxlA0uESCB0TxjAwQmfn1H+dUhPeLlbNimHlH0/hXPd
ifuNGozzADIRseQDyzjl8xGL1qRZLD3cfmda6RyZ+S3dQRuaRrcFCDccpY/pO+F8
jbx64zyqqNs+KV+SkQGOcKFhWTZGCfQ/zMDtDmQKjb3eTAkv1zdEOMw9zEjjmS0q
8FNl+2wO3VnvXwvBbtDdVCIaIq+jVcsy5XtnnV+bJ19Q9yue/XvCwWUEGAEKAA8C
GwwFAmEyoZoFCQueVRUACgkQrBB7OGaS2t1uHBAAhOYVvrtchRmzCvdNER1DtkIs
bgQPJ9OxbyfvmvoD06qxH7PrycLZKbt7yYpAUU/CMc86GwaEe0I5Nm1CTs6NvDIv
g3e7EPIS859tyQflbM56NlwbsopCuoCJYknuroIf/M6dW6vJKNXLMmnL/AtalUBw
X+5pblmGUUJep49oTOxQEnvnuqyvaGjXgFXix5PVFJD2ed5NnQeFpvfCpc/ioNOj
z7ORO82j1ht5nWqPraXX5AYhQFM/kwR1cK4LV7gVDd/q+dfGYHzpxQ/HtyX/Lasi
N6I52QqA95SM1ZZLPFLaNh6EvnB7uC9pLCYS8nvilX7/cez5PFff1e1gXCOT0jv3
mJ2exLmXV0BbfKgjccFCxhrdRLtukfiDfJkySy1zdscnpfng8wJ3xKRv43cUTz7M
Z24OYNMqK26aJZVXEQUYjCwsBylY/F5wjYAwgwZ8yF5RFix28P/K8JsIHb3QrAJK
sNWQAb03ZWis3N3spR5M9Mw3VuDZ3WUXq7mxB5M3kpVoZ3vETU5cwTbADYNPf4Sw
BDK2uIVtxabezxSBtz0FcyYoF+OW8q7r4WvoyC9/+3GfnozZLJcEIVDk4W2pMW4A
UhG/6drKTm3HkSDWIDu7d1sHWMffLEYfUHtN5DKkDkGoPfHvZvu9teR5yLfUrPTf
ktihPn/JMrmwa9pwi8LCwWUEGAEKAA8CGwwFAmCavPcFCQsGcHIACgkQrBB7OGaS
```

```
2t0uaA//UWRaRiHEAKeRqBG/T2ak+XZJNu7QHfNgoUEAub9Zru8oPPXx2AJLcHEN
KWmeFlLxADdWOZs4Bm9oOew3VQnR/dBqjnXfob9Rc+eYUjA3rXazM/QrqcU8Syi3
MjNGUmjdL5aQF+IppAMgOBLG1TEnM7C5/PvrGJuYpGEnkKEwMK/GYhqg2V60pHEV
Pvs66mefJpCzbZSy56qtknSt6yBNWc14XgDX6VTn2kW4CV/3vVJUuvjvYs9SPyY8
mKEXa6QvUd3PcXv6RiWk4lGYuT1+jh2VkcFQ+JnUwv9TbKFB9b5jq1bvW9+LMDEl
YXux7pBP5RPk+OLpyiExIRFWhi3x7aMWOzQ+I9yuNTeYkTHiEAQRUhs/1Fh4oLgI
v9QZgC0mRSN3zm8plQdivs1ZlAosAqqkA9BQwqsgosQe7P92irYIJqay0si9wGCD
wSMsmeXdIF6wW3/UMJZl66aarPeiZApGX0QdTZwjMh/QK/8gTKyeZulKmNkNfwWq
O17OirWqLKssVHtg3VUM8EIdh+oNqDDXSeWtYUmpPpWp+yWZ0x1MFFZhUQHQZTGu
TIj4A92LQzbrfj/jXRvWm2SrJMivUoiDUn+qxKIpVwFlI5gVb+uyTFhw89PCkphr
JwRi052RLoU9yd6Ek46UH4XfZZWrZuzY+zzB7oqGONphLgi/h3DCwWUEGAEKAA8C
GwwFAl771b8FCQlniTUACgkQrBB7OGaS2t2/MxAAjoEGPdzavhsOlXdPCRd1D5QJ
r8T/NSEV2z1cp8ZvdrkjNF09TBP4qsBnKJiuvY1Iw7OGX9W2okvXxgJizE45v9MH
WEMz4hmIjmAfRwcqENgpOc1IY/T0/+kkCW8dB6d30J1kT0n2PCRzN9L5vPqZXGTG
mLvd9MOjH1256w4uxLb+e1HMDTCqEN1ppq9G+EAR/29q8JZWs1marbZZWxSWcg/E
1YYbNafzklgjq4CLh/j8AEWSvLr39zRy9uvQ/yqAKZ4K4aZfh/SPupGDvsD6ZK54
EPHxErQ7aiXTbUHtvwhxWLOP6WmxFA3Shr6L6YUb6jq+0PVliFC517g3mxFHJtwt
yXGNIKhmzmr0l9OlsHafulJ/9QPfK3Ce32SkPhW/11MYA8HzduMv5Arp7cBczXSP
EUTmNIVKv3gTjSQrzRhwhHmMuqyDZ/rXQQ1jl2sxIDjO4MUMvVjYKF+OCNm42gVs
8ca3/wN9ZNU6hyFWeKQDuCAqPPbT5GO/DKseFEwB+07wwyH1RXbyl0v4fneg605X
S7lqhNtw2p1hDL0HYHDiV+aPZ+LoOmX6+dmnqE6bQJaIlVb922KWmliO7F3DkqP7
0jFlhoE1gfiXWkxP4Gy8wOobNfEMgvz02djkGQy+oQqeNdIcZFZgzPTGKB/nVgpt
9CcRDWjPltFCd2e1FBbCwWUEGAEKAA8FAld1gAUCGwwFCQeGH4AACgkQrBB7OGaS
2t1PIQ//Qc5VYfBCxpaMysaPQ44wXPEZSjxIGZhhMGzb1UzzAEYOw+RgKN5nNTXq
L2KoOkOrGnKqZOKByMdXwIPH/rGwwEsbbIpopnibf5ic5B/+xCTIK+qLIwX2ZLuk
NhbL6Y+E+7DxMMh+KqBWHONKkgwVY+rFWOfoops839ABKvc9/Ry4/qqkcb40AzpD
l1iQJ5vK/DMuaDWxWeKXqJLIl3WMGPcPfheuBZL1u7LEEHYKMgzvpbF81WIn3MBo
8jvxf2/o+kMafSSDqgvOu6yu8GOhmScpCbRJn7jV/HrG+tM+zy48TN6/MkGWSR7q
TD34pqBjyatVfVl6dGD6xj/i/Emt5hZB6qXruCDH7AWMoNx+FkDubs4sc4PKysZU
Itya6KdQFo2UeYsNwZhdn6QwKhd85um4JUHJCY0mARvjsQgWXH/5MR40cow77bbE
vVq0XNd+QRVlyT42CEtnIUOFLeDVuZrum5Tuvvna6ImMDoi/z6QcNeL79XsY2m6I
QVRiHr1BDb/8JLkfnWiwL8GRv169Kf8unx0y5u1YBpcMYkyDD2+pnnk3TY0rR+8X
8goecaS8fbyu/Q48K85ZMD8wKW/bzLQ+tK9y8xed24u2QERftMhIw9b6f45Nrrf/
PhgV8RnuwUusSbdDe8kw3eYTmLdzD4kZc9K7SdO2CqT+hm//9JI=
=uGHC
-----END PGP PUBLIC KEY BLOCK-----
```

Kunci sebelumnya

> **Important**
>
> Kunci baru dibuat sebelum yang sebelumnya kedaluwarsa. Akibatnya, pada saat tertentu, lebih dari satu kunci mungkin valid. Kunci digunakan untuk menandatangani artefak mulai hari pembuatannya, jadi gunakan kunci yang baru dikeluarkan saat validitas kunci tumpang tindih.

Tanggal kedaluwarsa: 2025-10-04

ID Kunci	0x AC1 07B386692DADD
Jenis	RSA
Size	4096/4096
Dibuat	30/06/2016
Tanggal kedaluwarsa	2025-10-04
ID Pengguna	AWS SDKs dan Alat < aws-dr-tools @amazon .com>
Sidik jari kunci	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 6692 DADD AC1

Untuk menyalin kunci publik OpenPGP berikut untuk SDK for Java ke clipboard, pilih ikon “Salin” di sudut kanan atas.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Hostname:
Version: Hockeypuck 2.2

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
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJnABcIBQkRa8qDAAoJEKwQezhmktrdl1MQAIwEuDar3OTxkfTa
cPNKDnNzxaWqrxZ3FTQ+PyrHhQ6usxxrvDKJS+uCjE9bmWHVFU1R4yQNF+72lJdw
5UhX0u+ZgT9afApE65uAZuwLhPsz8upXT8C6VeKXh3shdw7qXi2hrwtM1a0Pls4O
Cs2C9rLUDMJTySrVDDVwpnaAB+8DcFrs9bIt5Q3gdOUatdzDvcb7QKh9jUvzCpbE
cInb1epDN5MRzowMR4iU2VV1RzLxCvm7CQSyXfgf0DFLkXWiknhOq9eINmytJFG/
ntFdiZFkNZ5hP709loywdfNrmqB6PsF8BPGFh8gKw1pjowrfHpv6cNIqShmA76LT
3OHViOlqGFB7obffq//eZGPR0oYJFDrOdD2CFRoHnP3N++AfkA4SRN7eXwyoz6Pk
Do9WNIEEkAcp6PGvv7AokogDo/4OqmxgC6fN+3BTOstWpv4FlD4NxOZWsTs49wxg
kPlCCVf8t75aZZkcjXng1eClZZQ5SBlRtSB7gMqtP7MIn2J5w8spNbs5xQvJc76u
NvzwEasPkY+UcHd05RddOUwoKqDerLUG7YqdlNCJoQR1mBIgZButbQlMyaZcmQqO
iR0kwDi9h6Dl6fnUb2dFCNJw+eDHvsjG8HI3IVZMl0bUQ2kmmwr1O2YQ1ynJQm0l
lMllI4hFU8/lHNHm8ie5darpVXQEwsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiIFCQ+P/Z0ACgkQ
rBB7OGaS2t3HVg//S+/KbeOBf+nCdHsrWtp9kxvWIpAGvQhIbxx1tp/impfm+5Rm
fKPD0KX+g42fuMmOdDE4gjO4GjGd7ZY3bx+0zbDSdVebzmYCbPZ/BDP990oPKidd
w6Gl8PaIyqfuARKOESBETvAwNgwO4t2ocjs4pYZV+CuHvESYpqkuHjmHtye6ajZW
Mv24NhjVo4EfP33dPugTjXLjeuGT7qQpsYV3a66juHmPVkXwuPqxh9wTNc5TU6FG
UPSfIGMPL0xha7Rg2i5zvRaAxx4bHqGO8IAz/l/E/tJkV5xnt494HQam9UDbiFI0
Q0TSve1R6S45/UjQW6cycyduHtk72s9ipa9YM0ilTdLgKMWFjzYv4h4qeYvLw3oB
JGQew+I0I4dIrwL/TKet33EuFfwmyT9MaJBhqV6geFaQ0uVmwvzpAcvxIoSqSpkJ
B/kASqCEM/o0QZLuWc56cDsmMisD0ouVPt+c1Zk7AWLlf6j8LKYTbK0QxLRh/eeZ
jhSf3HnpaCfonbOoHmeo5d/o3EZ0AiA4GbT3xgScoIgXOT7KGgOWmtWkdyd3Zvl/
o6q6Hwpg4RsqCkwnfNm5ZIvmTAYXWc2hiICSicxrP0fek5Cc4xVgJR5RMNGyI7+m
ut1SE2WvLhMCwEyl5ecWF0tUze8VBlWkHJp0Y4k2ado39Zq/DZrTRQYEVrjCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmEyoZoFCQueVRUACgkQ
rBB7OGaS2t3axA//dgcz5T4Fs7LWdIQB/KRnvX64IzaGQcwt3FBhYH+sFaSaD+lu
752vi3jlGxMBKs2NFxk6e2U8xBEir3vfKYd+mZ5L6egXC9MYfvMO/UFEFH3A+t3V
dTOJK4RQcaL9+rFRVdDmZuifN9OFfy25d66JCZ5OiqgHTQViVmbrGw2cQdyNWXrq
YLgg+gktadkmzis4J6hF/le8NOBfrG3n+QthFl/v2ppYYW9pmmxzUIf6tAlR1Vr8
PhPukjuFfrPLRL3XPiK4LkdlSI5MX7Llq4RkcZN1NY1LWS8699wJOLRcr8aQYvzZ
Jm7tfZUaekJ5ScXJWJEaWT4poMbWxXINj6VwE+DqKWvjkzoxBXSIdLk4XThA1dIq
3n5SfMkfuWV2A2xHqJtEzI1XeTm/d/JRxIG8hjIs5FNMGJUSNANJuTVA2putCVf0
JbP4QlafXoEVOYwW/EFJY+brjQadwc/knf/QxszDcKb3THuTxR80A0h6ZysmtLEg
PCaD1xUCDdr4P7DGOtV7yMaDR6O8QKmb7TKzCKCPnHouqPT0DhSB2MRq437+mfSe
EGga/sPMxe0z8ugO2CnrCf3eplU3ZOy4eeQSThTKe0Hp5YlsF1cdZSvjt7GJaHR2
```

```
9A22nAJY9PojtlM+0Dkr/PH6r2brv3sEuACRNhzqCWhAve+zVnVLeb+Fk1rCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmCavPYFCQsGcHEACgkQ
rBB7OGaS2t3fqg//asHYSTBI4IoLibSF5ZJ8NaLo4EqgRtsO0W1zr8fCoFbxI+yI
qWrNXR7eoFj8tWO7Tj6S0kQr4QZcucLeILfox6CtDzO3f3WQTH9m/0si5U4Jf3RA
gBd0vwxVSSSNEsIUUfylOBHLVwlhaTVfh1h1dZhzb18LA1Vqu0100GGxvaG3dUl4
oMSRSK8EeaSO4hIM8ScjVyhjsuPRm1jlwd1EXZ5mkHRGRMGmwi3XysJjIeQRFmuG
a9uPes7I/K85r7X91BLz+G/mkSZsrHxzLxF0mSZtQdhq08GyMeQlJavs7sb1UVbg
ag30JEAjgqRsNIQOMIvl2/amrRJ7DUyQu5YkZQsAyCIhSVZw6zlJlkYVKSw1Cu1I
VX1n/Aahx5wwaQZA1dDTolX8WvL9O/KmEGWbKUSov40uoRwSOeXP3QhW75kD4zT0
n3pSUc7tXka8GAz5Ar+r90l4wc9as2WjWADo3CX8cF0zQOrN1naMQ++xBIAG9ms8
y3L4ECoRqvjCpjUAfhflxwSM7uclCgFBINFKSLV+QGxzWfHjg3+bSQd0hrRn9j4z
Di9aJmFESQ6iykbUjiUFrcI1NUCFKz2awaxh+Sq8UkYcvuxljxdK/zYYEG8DSdQ2
uwlssHCjYVpAb16hq1EAaSqnhv86020G4Z6JCg3AUZt9RlMBpK2Pn/NmPSzCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAl771vAFCQlnimUACgkQ
rBB7OGaS2t35Jw/9GhpQquJVUCAsc4az7+ad8q2d9OUKXOLl2x0j3/ofS8umZJYr
8KXZxPqBqvLjlUyAB5Ur4fWVBdp9iP4Vj74jmRih7YatK8heGmNoUAnI1/90qV5O
ypFObfvWLxvcUNizmS/LYKdNuYcHwyw4bFyTz9gd3XH6Dxak7YiAXz3JgJIXcIB3
ELfpsduzpncLeVwz/dKhYX5iJ7Y/xd4Ew8Ian8FyNDNRwcXobTpPAMNiGLG5xSLq
8RgQfkm07BVVDtu5tPw4V46gnBXGXNjPhSirGMoNbKZqtPO7TcBQhPQ9c95x73hs
kAqZKHrwi1B7cfy1I8y5sRFbPa0RXeVWQKEMZdwLsFhCbEex6iXHP+rMK0nSJf0G
9lF2eJk140n8K7wzakOnjvN00VfN/9D+xD2lCXCzbYbaDXX2tYOd4GS72fW9M/zT
96tIH5UtMGMOICuUJjnL34EbzBuxwTEnJKhDGQH2P+eUzM9jmuCTRLLGw2P0Zuof
6GsSNLf+5pw3BIvEZw5PYT1N6i3ml9MQ7p9BWr7flCF8CU/xzyPN7TVb/677Be7V
SL1rNFLNi0IdqcbLJ63pTWaUGkCSqRnMfq3cIDlzNz2LoVvOO8VVmdtFRkhHN8hJ
h7CUXlaqBOfaJhV3FqA9G8sXmSN3r3pusZb13kfCKsJUZorThWxdaUBuUH3CwX0E
EwEKACcFAld1gAUCGwMFCQeGH4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ
rBB7OGaS2t1aKA//dBaXto7zUFRbJvLgcSE3hJ0ChYZj8Uuo7iYK26mVycyBFQJK
Iv2XYFIJMwK1Rx1Ja1jA6BIKE3aYyx2LVTYU1Hke826dhH+kV2qN9C6t/VmYpV4b
n/i64po7EzD17ykrm5gB+z47uCvyC79/r8Ouns1mTf/JUql/hYJj2hf1MDWr07/X
Wc1zBLjlT93tg/43Vgz1wFdrUOcNx1+bQGXKTOi4AXSp3UvCL+2YsQ2/JsPf7ZzZ
awSuUVkZJrllp87S1MhZeHmTrCHkV1FHJyudLJErcH0337Jpd8xDRek7Klrx2pAS
cKIiyeAMIk/G1OuExYqEEVFQzMr9ZlMNuhNBJAMr5hVGsTgrwy2hlIrBktXav07d
OleSlsqw9ViZ7F6t90x5l+NkwXVsLsGYSzuNyIfomNuoCgA+cfM3TjzVp41qsg1J
WJc9Bavaan2pKF6Lb9Fq8u3HZk2u+YZbvZkqkXwTdZZQOkEmoVV4Y1G86bdpmPyj
8eV7C02NxPii4l+qV8qJQu/6DsA0QwMtBMUNODm3BF2+ZmUHuhMGxq9/4vDE8heE
ffYhHtNftV6JwwzGZmeZkrYA1P9AGLeVp/6iNUe8H5/oPvh4s2rRmqN+L/dQU1ix
iOAT5iAKoRQGkduXrWc4fAY5KxDB9qna4oqXO6QP8rEflI8ELcNgYEj4oJvOwU0E
V3WABQEQALzM0Cs9Zvd08xOEvbEBj59LrS9d0HVKQ61gmkNakWC+jR35VD6FXpe6
UYAcBLrEbVYfKw9P0p6MhFKAsb570JoznKGzE1rVYUZQzhD0RKje35rvkajvEcjG
AWMLTjr87pWHeD0389ER64bzORncfa/l+YP56PI+CThb2wUvTTONGJkPQUpVhH+P
256cQL/Y0Fwu4XLerpwN+YKgMQ47raRcydobPeSfMQr9fVKRyOzFEOrvNpCVDUqi
77d0gLDLjHlIlDyOX5554S8XYLb91eYOiFvnu2pTCKiiExRCSYK29mAQePKlTCCn
QxOjbmBbGS8mVIkpQ5vpvXvzpY3JIjMXaDGqWSQSYGXhECyxCR5eOtKYbCwwPIc2
rIl5gW6yXyw9pKmj5XafTP7YHTvRSr7CZ/VLkDkWl6AfQ9nPOg1mjwjpDFpmN71h
```

```
JlSKMaZkh0QGV5FW3dK+GLwxiWdqx3htbZErWyvumWQF/xBF7puKJBEXcoM5KfkJ
uZekBwcnVkfNFF2RdkM1ALq8InGzLXc7ROuEm0BXVirfju7JRtWLb3UhJWCuhRW2
muyYegSTkag5MduD1IJK37GL8WIlAL65taYgZegUoxHdSaEOefOhspxuduz8d33z
UV1WCFhi+r/+BMCQmTRbF8ao7fTC1dGd084DRP6qE/dMT4u0ZEn7ABEBAAHCwXwE
GAEKACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCZwAXCwUJEWvKhQAKCRCs
EHs4ZpLa3XtzD/9dwi1qffV70UTq8w/21jn1owHp09jxP7WHTmPWHE0BW5yFIWlV
A1gKN6Ym0dw+LvS5WOKJaRnyewUyBxWvZsn6Wlb5qzY7nmCOKJpYtuCUPwiqjXWP
EM8c/v0MojSuwMOXBAViLvOFhgdUrHn1lk962XvWAW++4DXFh2deaV0163IFMRmO
PNPDAiPWBVqvBANIh2sLRZ5gd1BXwpVrd+x8tzyr69YrN7hutPlCyPEUM9//mcEh
vFPsbW/iOx/foCE3NXhQm/rSMKecVn5csXBV2JOlMzi+8txYNrSBLkjbSB1AvTQ1
aG3+nCNCgM2XDLyoj0IrgZ1To4Ay5gmTOR+msY/cfoIuKFYenmtxy6jM8o5uSZHg
hoClrx9IA98hhGQ73G2r5EDpXuU/uCXn53Sswj65bl9IssfqEIoji/FonkkpEgeg
bGXFDUnrhicDO/WOzqpXf2Fa0DQWY+Vc/pt52ftBFgwzCNIUYDKUhCHPnZ0wtLtd
N2fkXHNiCavCDZlOud7FHHwmRNdj2q1uKxe4m+pFYmKwAU/H+Htkz9Gjsj+ZKedY
nnfai2s2gQOrbfwvV9VdhCWSuLK17ZnGTtiJuOUQIlV8n6QQJpohd3mVgmynu6gQ
uKw0YS2RuEUFv0vOg2tASA+4EM/SBUpGhudODLA4b5wO4gKmh1B1HqQrIsLBfAQY
AQoAJgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEokBQkPj/2fAAoJEKwQ
ezhmktrdwMAP/RpFylIL4yhgscBOEnQ7e3No8OraNk0z/YhSd125N/uQVEU94JGQ
rrvQ+4Lfve2laPweBDO18/A0CsmOyHPVQMA0a2vx8ItVdIcNc8iFkP4AJ1922lOq
i0Vh0b1UeZnlfK9+Qvq4PQ2lhWJr0uzyL/S38REsAT1I25sfJOP+RCaR1MH9dm85
E56Lee6uZR8SkGuiL6kGpPh6fWTNij3bICjth1iSSCL2HCOW8lvcwSldDu2EfILU
QCSqfSG7bF8dFk+nKhzhVXOUks3XGjLdICxZewU5ycryitpfRgARgZs2A43gshdi
fiKaX6Ksan03uhKDrLhDHNj2y07PUrFo8ggtlRpV/PrlB/UqCsC9FUOixbD+n4ZF
Sqov2qwelLj0f4mZ6yiLsTDUOFPrdkOlHTJZl7AF0zXZMM6CvaCUaJCKx9GVdSrR
+LI4wLQonPrTnXavhkC4intlqSX8ZQNLhEggdE8YwMEJn59R/nVIT3i5WzYph5R9
P4Vz3Yn7jRqM8wAyEbHkA8s45fMRi9akWSw93H5nWukcmfkt3UEbmka3BQg3HKWP
6TvhfI28euM8qqjbPilfkpEBjnChYVk2Rgn0P8zA7Q5kCo293kwJL9c3RDjMPcxI
45ktKvBTZftsDt1Z718LwW7Q3VQiGiKvo1XLMuV7Z51fmydfUPcrnv17wsFlBBgB
CgAPAhsMBQJhMqGaBQkLnlUVAAoJEKwQezhmktrdbhwQAITmFb67XIUZswr3TREd
Q7ZCLG4EDyfTsW8n75r6A9OqsR+z68nC2Sm7e8mKQFFPwjHPOhsGhHtCOTZtQk7O
jbwyL4N3uxDyEvOfbckH5WzOejZcG7KKQrqAiWJJ7q6CH/zOnVurySjVyzJpy/wL
WpVAcF/uaW5ZhlFCXqePaEzsUBJ757qsr2ho14BV4seT1RSQ9nneTZ0Hhab3wqXP
4qDTo8+zkTvNo9YbeZ1qj62l1+QGIUBTP5MEdXCuC1e4FQ3f6vnXxmB86cUPx7cl
/y2rIjeiOdkKgPeUjNWWSzxS2jYehL5we7gvaSwmEvJ74pV+/3Hs+TxX39XtYFwj
k9I795idnsS5l1dAW3yoI3HBQsYa3US7bpH4g3yZMkstc3bHJ6X54PMCd8Skb+N3
FE8+zGduDmDTKitumiWVVxEFGIwsLAcpWPxecI2AMIMGfMheURYsdvD/yvCbCB29
0KwCSrDVkAG9N2VorNzd7KUeTPTMN1bg2d1lF6u5sQeTN5KVaGd7xE1OXME2wA2D
T3+EsAQytriFbcWm3s8Ugbc9BXMmKBfjlvKu6+Fr6Mgvf/txn56M2SyXBCFQ5OFt
qTFuAFIRv+nayk5tx5Eg1iA7u3dbB1jH3yxGH1B7TeQypA5BqD3x72b7vbXkeci3
1Kz035LYoT5/yTK5sGvacIvCwsFlBBgBCgAPAhsMBQJgmrz3BQkLBnByAAoJEKwQ
ezhmktrdLmgP/1FkWkYhxACnkagRv09mpPl2STbu0B3zYKFBALm/Wa7vKDz18dgC
S3BxDSlpnhZS8QA3VjmbOAZvaDnsN1UJ0f3Qao5136G/UXPnmFIwN612szP0K6nF
PEsotzIzRlJo3S+WkBfiKaQDIDgSxtUxJzOwufz76xibmKRhJ5ChMDCvxmIaoNle
tKRxFT77OupnnyaQs22UsueqrZJ0resgTVnNeF4A1+lU59pFuAlf971SVLr472LP
```

```
Uj8mPJihF2ukL1Hdz3F7+kYlpOJRmLk9fo4dlZHBUPiZ1ML/U2yhQfW+Y6tW71vf
izAxJWF7se6QT+UT5Pji6cohMSERVoYt8e2jFjs0PiPcrjU3mJEx4hAEEVIbP9RY
eKC4CL/UGYAtJkUjd85vKZUHYr7NWZQKLAKqpAPQUMKrIKLEHuz/doq2CCamstLI
vcBgg8EjLJnl3SBesFt/1DCWZeummqz3omQKRl9EHU2cIzIf0Cv/IEysnmbpSpjZ
DX8Fqjtezoq1qiyrLFR7YN1VDPBCHYfqDagw10nlrWFJqT6VqfslmdMdTBRWYVEB
0GUxrkyI+APdi0M2634/410b1ptkqyTIr1KIg1J/qsSiKVcBZSOYFW/rskxYcPPT
wpKYaycEYtOdkS6FPcnehJOOlB+F32WVq2bs2Ps8we6KhjjaYS4Iv4dwwsFlBBgB
CgAPAhsMBQJe+9W/BQkJZ4k1AAoJEKwQezhmktrdvzMQAI6BBj3c2r4bDpV3TwkX
dQ+UCa/E/zUhFds9XKfGb3a5IzRdPUwT+KrAZyiYrr2NSMOzhl/VtqJL18YCYsxO
Ob/TB1hDM+IZiI5gH0cHKhDYKTnNSGP09P/pJAlvHQend9CdZE9J9jwkczfS+bz6
mVxkxpi73fTDox9duesOLsS2/ntRzA0wqhDdaaavRvhAEf9vavCWVrNZmq22WVsU
lnIPxNWGGzWn85JYI6uAi4f4/ABFkry69/c0cvbr0P8qgCmeCuGmX4f0j7qRg77A
+mSueBDx8RK0O2ol021B7b8IcVizj+lpsRQN0oa+i+mFG+o6vtD1ZYhQude4N5sR
RybcLclxjSCoZs5q9JfTpbB2n7pSf/UD3ytwnt9kpD4Vv9dTGAPB83bjL+QK6e3A
XM10jxFE5jSFSr94E40kK80YcIR5jLqsg2f610ENY5drMSA4zuDFDL1Y2ChfjgjZ
uNoFbPHGt/8DfWTVOochVnikA7ggKjz20+RjvwyrHhRMAftO8MMh9UV28pdL+H53
oOtOV0u5aoTbcNqdYQy9B2Bw4lfmj2fi6Dpl+vnZp6hOm0CWiJVW/dtilppYjuxd
w5Kj+9IxZYaBNYH4l1pMT+BsvMDqGzXxDIL89NnY5BkMvqEKnjXSHGRWYMz0xigf
51YKbfQnEQ1oz5bRQndntRQWwsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQ
ezhmktrdTyEP/0HOVWHwQsaWjMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCje
ZzU16i9iqDpDqxpyqmTigcjHV8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF
9mS7pDYWy+mPhPuw8TDIfiqgVhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+
NAM6Q5dYkCebyvwzLmg1sVnil6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNVi
J9zAaPI78X9v6PpDGn0kg6oLzrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB
lkke6kw9+KagY8mrVX1ZenRg+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHOD
ysrGVCLcmuinUBaNlHmLDcGYXZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKM
O+22xL1atFzXfkEVZck+NghLZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7
GNpuiEFUYh69QQ2//CS5H51osC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02N
K0fvF/IKHnGkvH28rv0OPCvOWTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+O
Ta63/z4YFfEZ7sFLrEm3Q3vJMN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=bboB
-----END PGP PUBLIC KEY BLOCK-----
```

Tanggal kedaluwarsa: 2024-10-08

ID Kunci	0x AC1 07B386692DADD
Jenis	RSA
Size	4096/4096
Dibuat	30/06/2016

Tanggal kedaluwarsa	2024-10-08
ID Pengguna	AWS SDKs dan Alat < aws-dr-tools @amazon .com>
Sidik jari kunci	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 6692 DADD AC1

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

Halaman ini mencantumkan perubahan penting pada Panduan AWS SDK untuk Java Pengembang selama sejarahnya.

Panduan ini diterbitkan pada: 1 Oktober 2025.

Oktober 1, 2025

Tambahkan kunci PGP baru yang kedaluwarsa pada 2026-09-27.

Oktober 5, 2024

Perbarui informasi kunci OpenPGP saat ini.

September 4, 2024

Tambahkan informasi tentang endpoint AWS berbasis akun untuk DynamoDB. Lihat, the section called “Gunakan AWS titik akhir berbasis akun”.

21 Mei 2024, 2024

Hapus instruksi untuk mengatur properti `networkaddress.cache.ttl` keamanan dengan menggunakan properti sistem baris perintah java. Lihat, Cara mengatur JVM TTL.

Januari 12, 2024

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

6 Desember 2023

- Berikan kunci OpenPGP saat ini.

14 Maret 2023

- Memperbarui panduan untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi lebih lanjut, lihat Praktik terbaik keamanan di IAM.

28 Juli 2022

- Menambahkan peringatan bahwa EC2 -Classic akan pensiun pada 15 Agustus 2022.

22 Mar 2018

- Menghapus mengelola sesi Tomcat sebagai DynamoDB contoh karena alat itu tidak lagi didukung.

November 2, 2017

- Menambahkan contoh kriptografi untuk klien Amazon S3 enkripsi, termasuk topik baru: Gunakan enkripsi sisi Amazon S3 klien dan enkripsi sisi Amazon S3 klien dengan kunci terkelola AWS KMS dan enkripsi sisi klien dengan kunci master Amazon S3 klien.

14 Apr 2017

- Membuat sejumlah pembaruan pada AWS SDK untuk Java bagian Amazon S3 Contoh Menggunakan, termasuk topik baru: Mengelola Izin Amazon S3 Akses untuk Bucket dan Objek dan Mengonfigurasi Amazon S3 Bucket sebagai Situs Web.

04 Apr 2017

- Topik baru, Mengaktifkan Metrik untuk AWS SDK untuk Java menjelaskan cara menghasilkan metrik kinerja aplikasi dan SDK untuk. AWS SDK untuk Java

03 Apr 2017

- Menambahkan CloudWatch contoh baru ke CloudWatch Contoh Menggunakan AWS SDK untuk 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 untuk Java bagian Amazon EC2 Contoh Menggunakan: Mengelola Amazon EC2 Instans, Menggunakan Alamat IP Elastis di Amazon EC2, Menggunakan wilayah dan zona ketersediaan, Bekerja dengan Pasangan Amazon EC2 Kunci, dan Bekerja dengan Grup Keamanan di Amazon EC2.

21 Mar 2017

- Menambahkan serangkaian contoh IAM baru ke Contoh IAM Menggunakan AWS SDK untuk Java bagian: Mengelola Kunci Akses IAM, Mengelola Pengguna IAM, Menggunakan Alias Akun IAM, Bekerja dengan Kebijakan IAM, dan Bekerja dengan Sertifikat Server IAM

13 Mar 2017

- Menambahkan tiga topik baru ke Amazon SQS bagian: Mengaktifkan Polling Panjang untuk Antrian Amazon SQS Pesan, Mengatur Batas Waktu Visibilitas Amazon SQS, dan Menggunakan Antrian Surat Mati di. Amazon SQS

26 Jan 2017

- Menambahkan Amazon S3 topik baru, Menggunakan TransferManager untuk Amazon S3 Operasi, dan Praktik Terbaik untuk AWS Pengembangan baru dengan AWS SDK untuk Java topik di AWS SDK untuk Java bagian Menggunakan.

16 Jan 2017

- Menambahkan Amazon S3 topik baru, Mengelola Akses ke Amazon S3 Bucket Menggunakan Kebijakan Bucket, dan dua Amazon SQS topik baru, Bekerja dengan Antrian Amazon SQS Pesan dan Mengirim Menerima dan Menghapus Pesan. Amazon SQS

Desember 16, 2016

- Menambahkan contoh topik baru untuk DynamoDB: Bekerja dengan Tabel di DynamoDB dan Bekerja dengan Item di DynamoDB.

September 26, 2016

- Topik di bagian Advanced telah dipindahkan ke Menggunakan AWS SDK untuk Java, karena mereka benar-benar penting untuk menggunakan SDK.

25 Agustus 2016

- Topik baru, Membuat Klien Layanan, telah ditambahkan ke Menggunakan AWS SDK untuk Java, yang menunjukkan cara menggunakan pembangun klien untuk menyederhanakan pembuatan Layanan AWS klien.

  Bagian Contoh AWS SDK untuk 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 untuk Java bagian Menggunakan, menjelaskan cara bekerja dengan metode klien asinkron yang mengembalikan objek atau yang mengambil file. `Future` `AsyncHandler`

26 Apr 2016

- Topik Persyaratan Sertifikat SSL telah dihapus, karena tidak lagi relevan. Support untuk sertifikat yang ditandatangani SHA-1 tidak digunakan lagi pada tahun 2015 dan situs yang menyimpan skrip pengujian telah dihapus.

Mar 14, 2016

- Menambahkan topik baru ke Amazon SWF bagian: Tugas Lambda, yang menjelaskan cara menerapkan Amazon SWF alur kerja yang memanggil Lambda fungsi sebagai tugas sebagai alternatif untuk menggunakan aktivitas tradisional. Amazon SWF

04 Mar 2016

- Amazon SWF Contoh Menggunakan AWS SDK untuk Java bagian telah diperbarui dengan konten baru:
  - Amazon SWF Dasar-dasar - Memberikan informasi dasar tentang cara memasukkan SWF dalam proyek Anda.

- Membangun Amazon SWF Aplikasi Sederhana - Tutorial baru yang memberikan step-by-step panduan bagi pengembang Java yang baru Amazon SWF.
- Mematikan Aktivitas dan Alur Kerja Pekerja dengan Anggun - Menjelaskan bagaimana Anda dapat menutup kelas Amazon SWF pekerja dengan anggun menggunakan kelas konkurensi Java.

Februari 23, 2016

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

28 Des 2015

- the section called “Mengatur JVM TTL untuk pencarian nama DNS”telah dipindahkan dari Advanced ke Using the AWS SDK untuk 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 SHA256 yang ditandatangani 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 SHA256 -signed.

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 untuk 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 untuk 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 instance. EC2

September 9, 2013

- Topik ini, Riwayat Dokumen, melacak perubahan pada Panduan AWS SDK untuk Java Pengembang. Hal ini dimaksudkan sebagai pendamping sejarah catatan rilis.