Entwicklerhandbuch für Version 1.x

AWS SDK for Java 1.x

AWS SDK for Java 1.x: Entwicklerhandbuch für Version 1.x

Table of Contents

Wir haben die bevorstehende end-of-support für AWS SDK for Java (v1) angekündigt. Wir empfehlen Ihnen, zu AWS SDK for Java v2 zu migrieren. Datumsangaben, zusätzliche Details und Informationen zur Migration finden Sie in der verlinkten Ankündigung.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.

Entwicklerhandbuch -AWS SDK for Java 1.x

Das AWS SDK for Javastellt eine Java-API fürAWS Dienste bereit. Mit Hilfe des SDKs können Sie Java-Anwendungen erstellen, die mit Amazon S3, Amazon EC2, DynamoDB und weiteren Services funktionieren. Unterstützung für neue Services ergänzen wir regelmäßig im AWS SDK for Java. Eine Liste der unterstützten Services und deren API-Versionen, die in den einzelnen Versionen des SDKs enthalten sind, finden Sie in den Versionshinweisen für die Version, mit der Sie arbeiten.

Version 2 des SDK veröffentlicht

Schauen Sie sich die neue VersionAWS SDK for Java 2.x unter https://github.com/aws/aws-sdk-java-v 2/ an. Es enthält mit Spannung erwartete Funktionen, z. B. die Möglichkeit, eine HTTP-Implementierung einzubinden. Informationen zu den ersten Schritten finden Sie im AWS SDK for Java2.x-Entwicklerhandbuch.

Zusätzliche Dokumentation und Ressourcen

Zusätzlich zu diesem Handbuch stehen AWS SDK for Java-Entwicklern folgende wertvolle Online-Ressourcen zur Verfügung:

- AWS SDK for Java API Referenz
- Java-Entwicklerblog
- Java-Entwicklerforen
- GitHub:
  - Dokumentationsquelle
  - Dokumentationsprobleme
  - SDK-Quellcode
  - SDK-Probleme
  - SDK-Beispiele
  - Gitter-Kanal
- Der AWS-Codebeispiel-Katalog
- @awsforjava (Twitter)
- Versionshinweise

Unterstützung für Eclipse-IDE

Wenn Sie Code mit der Eclipse-IDE entwickeln, können Sie den verwenden, AWS Toolkit for Eclipseum denAWS SDK for Java zu einem vorhandenen Eclipse-Projekt hinzuzufügen oder ein neuesAWS SDK for Java Projekt zu erstellen. Das Toolkit unterstützt auch das Erstellen und Hochladen von Lambda-Funktionen, das Starten und Überwachen von Amazon EC2-Instances, das Verwalten von IAM-Benutzern und Sicherheitsgruppen, einen AWS CloudFormation-Vorlageneditor und vieles mehr.

Die vollständige Dokumentation finden Sie im AWS Toolkit for EclipseBenutzerhandbuch.

Entwicklung von Anwendungen für Android

Wenn Sie ein Android-Entwickler sind,Amazon Web Services veröffentlichen Sie ein SDK, das speziell für die Android-Entwicklung entwickelt wurde: das Amplify Android (AWSMobile SDK for Android).

Anzeigen des SDK-Versionsverlaufs

Den Versionsverlauf des AWS SDK for Java mit Angaben zu Änderungen und unterstützten Services je nach SDK-Version finden Sie in den SDK-Versionshinweisen.

Erstellen von Java-Referenzdokumentationen für frühere SDK-Versionen

Die AWS SDK for JavaAPI-Referenz stellt den neuesten Build der Version 1.x des SDK dar. Wenn Sie eine frühere Version der 1.x-Version verwenden, möchten Sie möglicherweise auf die SDK-Referenzdokumentation zugreifen, die der von Ihnen verwendeten Version entspricht.

Die einfachste Methode zum Erstellen der Dokumentation besteht darin, das Build-Tool von Apache Maven zu nutzen. Laden Sie Maven zuerst herunter und installieren Sie es, falls es auf Ihrem System noch nicht vorhanden ist, und erstellen Sie die Referenzdokumentation dann mit den folgenden Schritten:

1. Suchen Sie die SDK-Version, die Sie verwenden, und wählen Sie sie auf der Releases-Seite des SDK-Repositorys ausGitHub.

2. Wählen Sie entweder den Link`zip` (die meisten Plattformen, einschließlich Windows) oder`tar.gz` (Linux, macOS oder Unix), um das SDK auf Ihren Computer herunterzuladen.
3. Extrahieren Sie das Archiv in ein lokales Verzeichnis.
4. Navigieren Sie in der Befehlszeile zu dem Verzeichnis, in das Sie das Archiv entpackt haben. Geben Sie dann folgenden Befehl ein:

   ```
   mvn javadoc:javadoc
   ```

5. Nachdem die Erstellung abgeschlossen ist, finden Sie die generierte HTML-Dokumentation im Verzeichnis `aws-java-sdk/target/site/apidocs/`.

Erste Schritte

Dieser Abschnitt enthält Informationen zur Installation, Einrichtung und Verwendung des AWS SDK for Java.

Themen

Grundeinstellungen, mit denen Sie arbeiten können AWS-Services

Übersicht

Für die erfolgreiche Entwicklung von Anwendungen, die AWS-Services über die zugreifenAWS SDK for Java, sind die folgenden Bedingungen erforderlich:

- Sie müssen sich bei dem AWS Zugangsportal anmelden können, das im verfügbar istAWS IAM Identity Center.
- Die für das SDK konfigurierten Berechtigungen der IAM-Rolle müssen den Zugriff auf die Rechte ermöglichenAWS-Services, die Ihre Anwendung benötigt. Die mit der PowerUserAccessAWSverwalteten Richtlinie verknüpften Berechtigungen reichen für die meisten Entwicklungsanforderungen aus.
- Eine Entwicklungsumgebung mit den folgenden Elementen:
  - Gemeinsam genutzte Konfigurationsdateien, die auf folgende Weise eingerichtet sind:
    - Die `config` Datei enthält ein Standardprofil, das eine angibtAWS-Region.
    - Die `credentials` Datei enthält temporäre Anmeldeinformationen als Teil eines Standardprofils.
  - Eine geeignete Installation von Java.
  - Ein Tool zur Build-Automatisierung wie Maven oder Gradle.
  - Ein Texteditor für die Arbeit mit Code.
  - (Optional, aber empfohlen) Eine IDE (integrierte Entwicklungsumgebung) wie IntelliJ IDEA, Eclipse oder. NetBeans

Wenn Sie eine IDE verwenden, können Sie AWS Toolkit s auch integrieren, um einfacher damit zu arbeitenAWS-Services. Die AWS Toolkit for IntelliJund AWS Toolkit for Eclipsesind zwei Toolkits, die Sie für die Java-Entwicklung verwenden können.

> **Important**
>
> Bei den Anweisungen in diesem Einrichtungsabschnitt wird davon ausgegangen, dass Sie oder Ihre Organisation IAM Identity Center verwenden. Wenn Ihre Organisation einen externen Identitätsanbieter verwendet, der unabhängig von IAM Identity Center arbeitet, finden Sie heraus, wie Sie temporäre Anmeldeinformationen für das SDK for Java erhalten können. Folgen Sie diesen Anweisungen, um der `~/.aws/credentials` Datei temporäre Anmeldeinformationen hinzuzufügen.
> Wenn Ihr Identitätsanbieter der `~/.aws/credentials` Datei automatisch temporäre Anmeldeinformationen hinzufügt, stellen Sie sicher, dass der Profilname `[default]` so lautet, dass Sie dem SDK keinen Profilnamen angeben müssen oderAWS CLI.

Anmeldemöglichkeit für das AWS Zugangsportal

Das AWS Zugriffsportal ist die Website, auf der Sie sich manuell beim IAM Identity Center anmelden. Das Format der URL ist `d-xxxxxxxxxx.awsapps.com/start` oder*`your_subdomain`*`.awsapps.com/start`.

Wenn Sie mit dem AWS Zugriffsportal nicht vertraut sind, folgen Sie den Anweisungen für den Kontozugriff in Schritt 1 des Themas zur IAM Identity Center-Authentifizierung im Referenzhandbuch zu AWS SDKs und Tools. Folgen Sie Schritt 2 nicht, da AWS SDK for Java 1.x die automatische Tokenaktualisierung und das automatische Abrufen temporärer Anmeldeinformationen für das in Schritt 2 beschriebene SDK nicht unterstützt.

Gemeinsame Konfigurationsdateien einrichten

Die gemeinsam genutzten Konfigurationsdateien befinden sich auf Ihrer Entwicklungsarbeitsstation und enthalten grundlegende Einstellungen, die von allen AWS SDKs und der AWS Command Line Interface (CLI) verwendet werden. Die gemeinsam genutzten Konfigurationsdateien können eine Reihe von Einstellungen enthalten, aber diese Anweisungen legen die grundlegenden Elemente fest, die für die Arbeit mit dem SDK erforderlich sind.

Einrichten der geteilten **`config`** Datei-

Das folgende Beispiel zeigt den Inhalt einer gemeinsam genutzten `config` Datei.

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

Verwenden Sie für Entwicklungszwecke die Stelle, an der Sie Ihren Code ausführen möchten, am AWS-Region nächsten liegt. Eine Liste der Regionalcodes, die in der `config` Datei verwendet werden sollen, finden Sie in der Allgemeine Amazon Web Services-Referenz Anleitung. Die `json` Einstellung für das Ausgabeformat ist einer von mehreren möglichen Werten.

Folgen Sie den Anweisungen in diesem Abschnitt, um die `config` Datei zu erstellen.

Temporäre Anmeldeinformationen für das SDK einrichten

Nachdem Sie über das Zugriffsportal AWS Zugriff auf eine AWS-Konto und IAM-Rolle haben, konfigurieren Sie Ihre Entwicklungsumgebung mit temporären Anmeldeinformationen, auf die das SDK zugreifen kann.

Schritte zum Einrichten einer lokalen **`credentials`** Datei mit temporären Anmeldeinformationen

1. Erstellen Sie eine gemeinsam genutzte `credentials` Datei.
2. Fügen Sie in die `credentials` Datei den folgenden Platzhaltertext ein, bis Sie funktionierende temporäre Anmeldeinformationen eingefügt haben.

   ```
   [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. Speichern Sie die Datei. Die Datei `~/.aws/credentials` sollte jetzt auf Ihrem lokalen Entwicklungssystem existieren. Diese Datei enthält das [Standard-] Profil, das das SDK for Java verwendet, wenn kein bestimmtes benanntes Profil angegeben ist.
4. Melden Sie sich beim AWS-Zugriffsportal an.
5. Folgen Sie diesen Anweisungen unter der Überschrift Manuelle Aktualisierung der Anmeldeinformationen, um die Anmeldeinformationen für die IAM-Rolle aus dem AWS -Zugriffsportal zu kopieren.

a. Wählen Sie für Schritt 4 in den verlinkten Anweisungen das -Konto der IAM-Rolle aus, die den Zugriff Ihren Entwicklungsanforderungen entsprechend gewährt. Diese Rolle hat normalerweise einen Namen wie PowerUserAccessoder Entwickler.

b. Wählen Sie für Schritt 7 die Option Profil manuell zu Ihrer AWS -Anmeldeinformationsdatei hinzufügen aus und kopieren Sie den Inhalt.

6. Fügen Sie die kopierten Anmeldeinformationen in Ihre lokale `credentials` Datei ein und entfernen Sie alle eingefügten Profilnamen. Ihre Datei sollte wie folgt aussehen:

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

7. Speichern Sie die `credentials` Datei

Das SDK for Java greift auf diese temporären Anmeldeinformationen zu, wenn es einen Service-Client erstellt und sie für jede Anfrage verwendet. Die in Schritt 5a ausgewählten Einstellungen für die IAM-Rolle bestimmen, wie lange die temporären Anmeldeinformationen gültig sind. Die maximale Dauer beträgt zwölf Stunden.

Wenn die temporären Anmeldeinformationen abgelaufen sind, wiederholen Sie die Schritte 4 bis 7.

Installieren Sie eine Java-Entwicklungsumgebung

Für das AWS SDK for Java ist das J2SE Development Kit 6.0 oder neuer erforderlich. Sie können die neueste Java-Software von http://www.oracle.com/technetwork/java/javase/downloads/ herunterladen.

> **Important**
>
> Java-Version 1.6 (JS2E 6.0) hatte keine integrierte Unterstützung für SHA256-signierte SSL-Zertifikate, die für alle HTTPS-Verbindungen AWS nach dem 30. September 2015 erforderlich sind.
> Java Version 1.7 und neuer enthalten aktualisierte Zertifikate und sind von diesem Problem nicht betroffen.

Auswählen einer JVM

Damit Ihre serverbasierten Anwendungen mit dem AWS SDK for Java die optimale Leistung erzielen, empfehlen wir, dass Sie die 64-Bit-Version der Java Virtual Machine (JVM) verwenden. Diese JVM kann nur im Servermodus ausgeführt werden, selbst wenn Sie zur Laufzeit die `-Client`-Option angeben.

Die Nutzung der 32-Bit-Version der JVM mit der Laufzeitoption `-Server` sollte eine vergleichbare Leistung wie die 64-Bit-JVM aufweisen.

Wege, um das zu bekommenAWS SDK for Java

Voraussetzungen

Zur Nutzung des AWS SDK for Java benötigen Sie Folgendes:

- Sie müssen in der Lage sein, sich beimAWS Zugangsportal anzumelden, das in der verfügbar istAWS IAM Identity Center.
- Eine geeignete Installation von Java.
- Temporäre Anmeldeinformationen wurden in Ihrer lokalen gemeinsam genutzten`credentials` Datei eingerichtet.

Inthe section called “Aufbau der Grundeinstellungen” diesem Thema finden Sie Anweisungen zur Einrichtung der Verwendung des SDK for Java.

Verwenden Sie ein Build-Tool, um Abhängigkeiten für das SDK for Java zu verwalten

Wir empfehlen, Apache Maven oder Gradle mit Ihrem Projekt zu verwenden, um auf die erforderlichen Abhängigkeiten des SDK for Java zuzugreifen. In diesem Abschnitt wird beschrieben, wie Sie diese Tools verwenden.

Laden Sie das SDK herunter und extrahieren Sie es (nicht empfohlen)

Wir empfehlen, dass Sie ein Build-Tool verwenden, um auf das SDK für Ihr Projekt zuzugreifen. Sie können jedoch eine vorgefertigte JAR-Datei mit der neuesten Version des SDK herunterladen.

> **Note**
>
> Weitere Informationen zum Herunterladen und Erstellen von früheren Versionen des SDKs finden Sie unter Installieren von früheren Versionen des SDKs.

1. Laden Sie das SDK von sdk-for-javahttps://.amazonwebservices.com/latest/aws-java-sdk .zip herunter.
2. Extrahieren Sie den Inhalt nach dem Herunterladen des SDKs in ein lokales Verzeichnis.

Das SDK enthält folgende Verzeichnisse:

- `documentation`- enthält die API-Dokumentation (auch im Internet verfügbar: AWS SDK for JavaAPI-Referenz).
- `lib`- enthält die `.jar` SDK-Dateien.
- `samples`- enthält funktionierenden Beispielcode, der demonstriert, wie das SDK verwendet wird.
- `third-party/lib`- enthält Bibliotheken von Drittanbietern, die vom SDK verwendet werden, wie Apache Commons Logging, AspectJ und das Spring Framework.

Um das SDK zu verwenden, fügen Sie den vollständigen Pfad zu den Verzeichnissen `lib` und `third-party` zu den Abhängigkeiten in Ihrer Build-Datei hinzu und fügen Sie sie zu Ihrem Java-`CLASSPATH` hinzu, um Ihren Code auszuführen.

Frühere Versionen des SDK aus dem Quellcode erstellen (nicht empfohlen)

Nur die neueste Version des kompletten SDK wird in vorgefertigter Form als herunterladbare JAR-Datei bereitgestellt. Sie können jedoch eine vorherige Version des SDKs mit Apache Maven (Open Source) erstellen. Maven lädt alle erforderlichen Abhängigkeiten, erstellt und installiert das SDK in einem Schritt. Besuchen Sie http://maven.apache.org/, um Installationsanweisungen und weitere Informationen zu erhalten.

1. Gehen Sie zur GitHub Seite des SDK unter: AWS SDK for Java(GitHub).
2. Wählen Sie das Tag aus, das der gewünschten SDK-Versionsnummer entspricht. Zum Beispiel `1.6.10`.
3. Klicken Sie auf die Schaltfläche Download ZIP, um die ausgewählte Version des SDKs herunterzuladen.

4. Extrahieren Sie die Datei in ein Verzeichnis auf Ihrem Entwicklungssystem. Bei vielen Systemen können Sie dazu den grafischen Datei-Manager oder das `unzip`-Dienstprogramm in einem Terminal-Fenster nutzen.
5. Navigieren Sie in einem Terminal-Fenster in das Verzeichnis, in das Sie die SDK-Quelldateien entpackt haben.
6. Erstellen und installieren Sie das SDK mit dem folgenden Befehl (Maven erforderlich):

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

   Die resultierende `.jar`-Datei wird im `target`-Verzeichnis erstellt.
7. (Optional) Erstellen Sie die API-Referenz-Dokumentation mit dem folgenden Befehl:

   ```
   mvn javadoc:javadoc
   ```

   Die Dokumentation wird im Verzeichnis `target/site/apidocs/` erstellt.

HIER ungungungungungung

Die Verwendung von Build-Tools hilft bei der Verwaltung der Entwicklung von Java-Projekten. Es sind mehrere Build-Tools verfügbar, aber wir zeigen, wie man mit zwei beliebten Build-Tools — Maven und Gradle — loslegen kann. In diesem Thema erfahren Sie, wie Sie mit diesen Build-Tools das SDK for Java Java-Abhängigkeiten verwalten, das Sie für Ihre Projekte benötigen.

Themen

- Verwenden Sie das SDK mit Apache Maven
- Verwenden Sie das SDK mit Gradle

Verwenden Sie das SDK mit Apache Maven

Sie können Apache Maven zum Konfigurieren und Erstellen von AWS SDK for Java-Projekten sowie zum Erstellen des SDK selbst verwenden.

> **Note**
>
> Um die Anleitungen in diesem Thema nachzuvollziehen, sollten Sie Maven installiert haben. Wenn Maven noch nicht installiert ist, besuchen Sie http://maven.apache.org/, um es herunterzuladen und zu installieren.

Erstellen eines neuen Maven-Pakets

Sie können ein einfaches Maven-Paket erstellen, indem Sie ein Terminal-Fenster (eine Befehlszeile) öffnen und Folgendes ausführen:

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

Ersetzen Sie org.example.basicapp mit dem vollen Paket-Namespace Ihrer Anwendung und myapp mit dem Projektnamen (wird für den Verzeichnisnamen Ihres Projekts übernommen).

Erstellt standardmäßig eine Projektvorlage für Sie unter Verwendung des Schnellstart-Archetyps, der ein guter Ausgangspunkt für viele Projekte ist. Es sind noch mehr Archetypen verfügbar. Auf der Seite mit den Maven-Archetypen finden Sie eine Liste der mitgelieferten Archetypen. Sie können einen bestimmten Archetyp zur Nutzung auswählen, indem Sie das Argument `-DarchetypeArtifactId` an den Befehl `archetype:generate` anhängen. Beispiel:

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

> **Note**
>
> Viele weitere Informationen zum Erstellen und Konfigurieren von Projekten finden Sie im Maven Getting Started Guide.

Konfigurieren des SDKs als Maven-Abhängigkeit

Sie können das AWS SDK for Java nur dann in Ihrem Projekt verwenden, wenn Sie es in der Datei `pom.xml` Ihres Projekts als Abhängigkeit deklarieren. Ab Version 1.9.0 können Sie einzelne Komponenten oder das gesamte SDK importieren.

Angeben einzelner SDK-Module

Zur Auswahl einzelner SDK-Module verwenden Sie die AWS SDK for Java-Bill of Materials (BOM) für Maven. Dadurch wird sichergestellt, dass die angegebenen Module die gleiche SDK-Version nutzen und miteinander kompatibel sind.

Um die BOM zu verwenden, fügen Sie der Datei `pom.xml` Ihrer Anwendung einen Abschnitt `<dependencyManagement>` hinzu. Fügen Sie dabei `aws-java-sdk-bom` als Abhängigkeit hinzu und geben Sie die SDK-Version an, die Sie nutzen möchten:

```
<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>
```

Die neueste Version derAWS SDK for Java BOM, die auf Maven Central verfügbar ist, finden Sie unter: https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-bom. Auf dieser Seite können Sie auch sehen, welche von der BOM verwalteten Module (Abhängigkeiten) Sie im Abschnitt `<dependencies>` der Datei `pom.xml` Ihres Projekts einfügen können.

Sie können jetzt einzelne Module aus dem SDK zur Nutzung in Ihrer Anwendung auswählen. Da Sie die SDK-Version bereits in der BOM deklariert haben, müssen Sie die Versionsnummer nicht mehr für jede Komponente angeben.

```
<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>
```

Sie können auch die aufrufen, um AWS-Codebeispiel-Katalogzu erfahren, welche Abhängigkeiten Sie für ein bestimmtes Objekt verwenden solltenAWS-Service. Weitere Informationen finden Sie in der POM-Datei unter einem bestimmten Servicebeispiel. Wenn Sie beispielsweise an den Abhängigkeiten für denAWS S3-Dienst interessiert sind, finden Sie das vollständige Beispiel unter GitHub. (Schauen Sie sich den POM unter /java/example_code/s3 an).

Importieren aller SDK-Module

Wenn Sie das gesamte SDK als Abhängigkeit aufnehmen möchten, verwenden Sie nicht die BOM-Methode. Deklarieren Sie es stattdessen einfach wie folgt in `pom.xml`:

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

Erstellen Ihres Projekts

Sobald Ihr Projekt fertig eingerichtet ist, können Sie es mit dem Maven-Befehl `package` erstellen:

```
mvn package
```

Dadurch wird die `0jar`-Datei im Verzeichnis `target` angelegt.

Erstellen des SDKs mit Maven

Sie können Apache Maven verwenden, um das SDK aus den Quellen zu erstellen. Laden Sie dazu den SDK-Code von herunter GitHub, entpacken Sie ihn lokal und führen Sie dann den folgenden Maven-Befehl aus:

```
mvn clean install
```

Verwenden Sie das SDK mit Gradle

Um SDK-Abhängigkeiten für Ihr zu verwaltenGradleProjekt, importiere die Maven-Stückliste fürAWS SDK for Javain die Anwendung`build.gradle`datei.

> **Note**
>
> Ersetzen Sie in den folgenden Beispielen*1.12.529*in der Build-Datei mit einer gültigen Version vonAWS SDK for Java. Die neueste Version finden Sie in derZentrales Maven-Repository.

Projekteinrichtung für Gradle 4.6 oder höher

Seit Gradle 4.6können Sie die verbesserte POM-Unterstützungsfunktion von Gradle verwenden, um Stücklistendateien (BOM) zu importieren, indem Sie eine Abhängigkeit von einer Stückliste deklarieren.

1. Wenn Sie Gradle 5.0 oder höher verwenden, fahren Sie mit Schritt 2 fort. Andernfalls aktivieren Sie die Funktion IMPROVED_POM_SUPPORT in der `settings.gradle`-Datei.

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

2. Fügen Sie die Stückliste zur hinzuAbhängigkeitenAbschnitt der Anwendung`build.gradle`datei.

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

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

3. Geben Sie im Abschnitt dependencies (Abhängigkeiten) die SDK-Module an, die verwendet werden sollen. Die folgende enthält beispielsweise eine Abhängigkeit für 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 löst mit den Informationen aus der BOM automatisch die richtige Version der SDK-Abhängigkeiten auf.

Im Folgenden finden Sie ein Beispiel für eine vollständige Datei `build.gradle`, die eine Abhängigkeit für Amazon S3 enthält.

```
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**
>
> Ersetzen Sie im vorherigen Beispiel die Abhängigkeit fürAmazon S3mit den Abhängigkeiten vonAWSDienste, die Sie in Ihrem Projekt verwenden werden. Die Module (Abhängigkeiten), die von der verwaltet werdenAWS SDK for JavaBOM sind aufgeführt unterZentrales Maven-Repository.

Projekteinrichtung für Gradle-Versionen vor 4.6

Gradle-Versionen vor 4.6 verfügen über keine native Stücklistenunterstützung. Zur Verwaltung von AWS SDK for Java-Abhängigkeiten für Ihr Projekt verwenden Sie das Abhängigkeitsverwaltungs-Plugin von Spring für Gradle, um die Maven-Stückliste für das SDK zu importieren.

1. Fügen Sie das Plugin zur Verwaltung von Abhängigkeiten zu Ihrer Anwendung hinzu`build.gradle`datei.

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

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

2. Fügen Sie die BOM in den Abschnitt dependencyManagement der Datei ein.

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

3. Geben Sie im Abschnitt Abhängigkeiten die SDK-Module an, die Sie verwenden werden. Im folgenden Beispiel ist eine Abhängigkeit für Amazon S3 enthalten.

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

Gradle löst mit den Informationen aus der BOM automatisch die richtige Version der SDK-Abhängigkeiten auf.

Im Folgenden finden Sie ein Beispiel für eine vollständige Datei `build.gradle`, die eine Abhängigkeit für Amazon S3 enthält.

```
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**
>
> Ersetzen Sie im vorherigen Beispiel die Abhängigkeit fürAmazon S3mit den Abhängigkeiten vonAWSService, den Sie in Ihrem Projekt verwenden werden. Die Module (Abhängigkeiten), die von der verwaltet werdenAWS SDK for JavaBOM sind aufgeführt unterZentrales Maven-Repository.

Weitere Informationen über das Angeben von SDK-Abhängigkeiten mit der BOM finden Sie unter Verwenden des SDK mit Apache Maven.

AWSTemporäre Anmeldeinformationen einrichten undAWS-Region für die Entwicklung

Um mit dem eine Verbindung zu einem der unterstützten Dienste herzustellenAWS SDK for Java, müssen SieAWS temporäre Anmeldeinformationen angeben. DieAWSAWS SDKs undAWS Unix.

Dieses Thema enthält grundlegende Informationen zum Einrichten IhrerAWS temporären Anmeldeinformationen für die lokale Anwendungsentwicklung mithilfe vonAWS SDK for Java. Wenn Sie Anmeldeinformationen zur Nutzung innerhalb einer EC2-Instance einrichten möchten oder die Eclipse-IDE zur Entwicklung verwenden, sehen Sie sich stattdessen folgende Themen an:

- Wenn Sie eine EC2-Instance verwenden, erstellen Sie eine IAM-Rolle und geben Sie dann Ihrer EC2-Instance Zugriff auf diese Rolle, wie in Using IAM Roles to Grant Access toAWS Resources on gezeigtAmazon EC2.
- Richten SieAWS Anmeldeinformationen in Eclipse ein, indem Sie die verwenden AWS Toolkit for Eclipse. Weitere Informationen finden Sie im AWS Toolkit for EclipseBenutzerhandbuch unter AWSAnmeldeinformationen einrichten.

Verwenden temporäre Anmeldeinformationen

Sie können temporäre Anmeldeinformationen fürAWS SDK for Java auf verschiedene Arten konfigurieren, aber hier sind die empfohlenen Vorgehensweisen:

- Legen Sie temporäre Anmeldeinformationen in der Profildatei mit denAWS Anmeldeinformationen auf Ihrem lokalen System fest. Sie befindet sich unter:
  - `~/.aws/credentials` (Linux, MacOS und Unix)
  - `C:\Users\USERNAME\.aws\credentials` (Windows)

  the section called “Temporäre Anmeldeinformationen für das SDK einrichten”In diesem Handbuch finden Sie Anweisungen, wie Sie Ihre temporären Anmeldeinformationen erhalten.
- Stellen Sie die`AWS_SESSION_TOKEN` Umgebungsvariablen`AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, und ein.

  Um diese Variablen auf Linux, macOS oder Unix festzulegen, verwenden Sie :

```
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
```

In Windows können Sie die Variablen mit festlegen:

```
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
```

- Für eine EC2-Instance: Geben Sie eine IAM-Rolle an und gewähren Sie Ihrer EC2-Instance Zugriff auf diese Rolle. Eine ausführliche Beschreibung der Funktionsweise finden Sie unter IAM-Rollen fürAmazon EC2 imAmazon EC2 Benutzerhandbuch für Linux-Instances.

Sobald Sie IhreAWS temporären Anmeldeinformationen mit einer dieser Methoden eingerichtet haben, werden sie automatischAWS SDK for Java von der geladen, wobei die standardmäßige Anbieterkette für Anmeldeinformationen verwendet wird. Weitere Informationen zum Arbeiten mitAWS Anmeldeinformationen in Ihren Java-Anwendungen finden Sie unter Arbeiten mitAWS Anmeldeinformationen.

Aktualisieren von IMDS-Anmeldeinformationen

Das AWS SDK for Java unterstützt auf Wunsch die Aktualisierung von IMDS-Anmeldeinformationen im Hintergrund einmal pro Minute, unabhängig von der Ablaufzeit der Anmeldeinformationen. Auf diese Weise können Sie Ihre Anmeldeinformationen häufiger aktualisieren und die Wahrscheinlichkeit verringern, dass sich das Nichterreichen des IMDS auf die wahrgenommeneAWS Verfügbarkeit auswirkt.

```
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();
```

Stellen Sie dasAWS-Region

Sie sollten eine Standardeinstellung festlegenAWS-Region, die für den Zugriff aufAWS Dienste mit dem verwendet wirdAWS SDK for Java. Um die beste Netzwerkleistung zu erzielen, wählen Sie die Region aus, die geografisch in Ihrer Nähe (oder in der Nähe Ihrer Kunden) liegt. Eine Liste der Regionen für jeden Dienst finden Sie unter Regionen und Endpunkte in derAmazon Web Services Allgemeinen Referenz.

> **Note**
>
> Wenn Sie keine Region auswählen, wird standardmäßig us-east-1 verwendet.

Sie können ähnliche Techniken wie beim Festlegen von Anmeldeinformationen verwenden, um IhreAWS Standardregion festzulegen:

- Stellen Sie dasAWS-Region in derAWS Konfigurationsdatei auf Ihrem lokalen System ein, das sich unter:
  - ~x x x x x x x
  - C:\Users\USERNAME\ .aws\ config unter Windows

  Diese Datei sollte Zeilen im folgenden Format enthalten:

  +

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

  +

  Ersetzen Sie Ihre gewünschte RegionAWS-Region (z. B. „us-east-1") durch your_aws_region.
- Legen Sie die `AWS_REGION`-Umgebungsvariable fest.

  Unter Linux macOS x x x x

```
export AWS_REGION=your_aws_region
```

In Windows nutzen Sie :

```
set AWS_REGION=your_aws_region
```

Wobei your_aws_region der gewünschteAWS-Region Name ist.

Verwenden der AWS SDK for Java

Dieser Abschnitt enthält wichtige allgemeine Informationen zum Programmieren mit der AWS SDK for Java , die für alle Services gelten, die Sie möglicherweise mit dem SDK verwenden.

Servicespezifische Informationen und Beispiele für die Programmierung (für Amazon EC2, Amazon S3 Amazon SWF usw.) finden Sie unter AWS SDK for Java Codebeispiele.

Themen

Bewährte Methoden für die AWS Entwicklung mit der AWS SDK for Java

Die folgenden bewährten Methoden können Ihnen helfen, Probleme bei der Entwicklung von AWS Anwendungen mit der zu vermeiden AWS SDK for Java. Wir haben diese bewährten Methoden nach Service angeordnet.

S3

Vermeiden Sie ResetExceptions

Wenn Sie Objekte mithilfe Amazon S3 von Streams (entweder über einen -`AmazonS3Client` oder `TransferManager`) in hochladen, können Probleme mit der Netzwerkkonnektivität oder dem

Timeout auftreten. Standardmäßig AWS SDK for Java versucht der , fehlgeschlagene Übertragungen erneut zu versuchen, indem er den Eingabe-Stream vor dem Start einer Übertragung markiert und ihn dann vor dem erneuten Versuch zurücksetzt.

Wenn der Stream das Markieren und Zurücksetzen nicht unterstützt, löst das SDK eine aus, ResetException wenn vorübergehende Fehler auftreten und Wiederholungsversuche aktiviert sind.

Bewährte Methode

Wir empfehlen, dass Sie Streams einsetzen, die Markieren und Zurücksetzen unterstützen.

Die zuverlässigste Möglichkeit, einen zu vermeiden, ResetException besteht darin, Daten mithilfe einer Datei oder bereitzustellenFileInputStream, die der verarbeiten AWS SDK for Java kann, ohne durch Markierungs- und Zurücksetzungslimits eingeschränkt zu werden.

Wenn der Stream kein istFileInputStream, aber Markierungen und Zurücksetzen unterstützt, können Sie das Markierungslimit mithilfe der `setReadLimit` Methode von festlegenRequestClientOptions. Der Standardwert beträgt 128 KB. Wenn Sie den Leselimitwert auf ein Byte größer als die Größe des Streams festlegen, wird zuverlässig vermiedenResetException.

Beträgt die maximal erwartete Größe eines Streams beispielsweise 100 000 Bytes, legen Sie die Lesegrenze auf 100 001 (100 000 + 1) Bytes fest. Das Markieren und Zurücksetzen funktioniert immer für 100 000 oder weniger Bytes. Hinweis: Dies könnte bei einigen Streams dazu führen, dass die angegebene Anzahl an Bytes in den Arbeitsspeicher gepuffert wird.

Erstellen von Service-Clients

Um Anforderungen an zu stellen Amazon Web Services, erstellen Sie zunächst ein Service-Client-Objekt. Die empfohlene Methode besteht darin, den Service-Client-Generator zu nutzen.

Jede AWS-Service verfügt über eine Serviceschnittstelle mit Methoden für jede Aktion in der Service-API. Die Serviceschnittstelle für DynamoDB heißt beispielsweise AmazonDynamoDBClient . Jede Service-Schnittstelle verfügt über einen entsprechenden Client-Generator, mit dem Sie eine Implementierung der Service-Schnittstelle erstellen können. Die Client-Builder-Klasse für DynamoDB heißt AmazonDynamoDB ClientBuilder.

Abruf eines Client-Generators

Um eine Instance des Client-Generators abzurufen, verwenden Sie die statische Factory-Methode `standard`, wie im folgenden Beispiel gezeigt.

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

Sobald Sie einen Generator haben, können Sie die Eigenschaften des Clients anpassen, indem Sie die vielen praktischen Setter in der Generator-API nutzen. Beispielsweise können Sie wie folgt eine benutzerdefinierte Region und einen benutzerdefinierten Anmeldeinformationsanbieter festlegen.

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

> **Note**
>
> Die praktischen `withXXX`-Methoden geben das `builder`-Objekt zurück. So können Sie die Methodenaufrufe in Reihe schalten, was nicht nur einfacher ist, sondern auch für besser lesbaren Code sorgt. Nachdem Sie die gewünschten Eigenschaften konfiguriert haben, rufen Sie die `build`-Methode auf, um den Client zu erstellen. Sobald ein Client erstellt wurde, ist er unveränderlich und alle Aufrufe an `setRegion` oder `setEndpoint` schlagen fehl.

Ein Generator kann mehrere Clients mit der gleichen Konfiguration erstellen. Wenn Sie Ihre Anwendung entwerfen, sollten Sie daran denken, dass der Generator veränderlich und nicht threadsicher ist.

Der folgende Code verwendet den Generator als Factory für Client-Instances.

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

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

Der Builder stellt außerdem flüssige Setter für ClientConfiguration und RequestMetricCollectorund eine benutzerdefinierte Liste von RequestHandler2 bereit.

Im Folgenden finden Sie ein vollständiges Beispiel, in dem sämtliche konfigurierbaren Eigenschaften überschrieben werden.

```
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();
```

Erstellen von Async-Clients

Der AWS SDK for Java verfügt über asynchrone (oder asynchrone) Clients für jeden Service (außer Amazon S3) und einen entsprechenden asynchronen Client-Builder für jeden Service.

So erstellen Sie einen asynchronen DynamoDB-Client mit dem Standard ExecutorService

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

Zusätzlich zu den Konfigurationsoptionen, die der synchrone (oder Synchronisierungs-)Client-Builder unterstützt, können Sie mit dem asynchronen Client eine benutzerdefinierte festlegen, um die ExecutorFactory zu ändern`ExecutorService`, die der asynchrone Client verwendet. `ExecutorFactory` ist eine funktionale Schnittstelle, sodass es mit Java-8-Lambda-Ausdrücken und Methodenreferenzen zusammenarbeitet.

So erstellen Sie einen asynchronen Client mit einem benutzerdefinierten Executor

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

Verwenden von DefaultClient

Sowohl der synchrone als auch der asynchrone Client-Generator haben eine weitere Factory-Methode mit dem Namen `defaultClient`. Diese Methode erstellt einen Service-Client mit der Standardkonfiguration unter Verwendung der Standardanbieterkette zum Laden von Anmeldeinformationen und der AWS-Region. Wenn die Anmeldeinformationen oder die Region nicht aus der Umgebung, in der die Anwendung ausgeführt wird, ermittelt werden können, schlägt der Aufruf von `defaultClient` fehl. Weitere Informationen dazu, wie AWS Anmeldeinformationen und Region bestimmt werden, finden Sie unter Arbeiten mit Anmeldeinformationen und AWS-Region Auswahl.

So erstellen Sie einen Standard-Service-Client

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

Client-Lebenszyklus

Service-Clients im SDK sind threadsicher. Um eine bestmögliche Leistung zu erzielen, sollten Sie sie als langlebige Objekte behandeln. Jeder Client verfügt über seine eigene Verbindungspool-Ressource. Explizit Clients herunterfahren, wenn sie nicht mehr benötigt werden, um Ressourcenverluste zu vermeiden.

Um einen Client explizit herunterzufahren, rufen Sie die `shutdown`-Methode auf. Nach dem Aufruf von `shutdown` werden alle Client-Ressourcen freigegeben und der Client kann nicht mehr verwendet werden.

So fahren Sie einen Client herunter

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

Bereitstellen temporärer Anmeldeinformationen für die AWS SDK for Java

Um Anfragen an zu stellen Amazon Web Services, müssen Sie AWS temporäre Anmeldeinformationen für bereitstellen, die beim Aufrufen der -Services verwendet AWS SDK for Java werden sollen. Dafür können Sie eine der folgenden Möglichkeiten auswählen:

- Verwenden Sie die standardmäßige Anbieterkette von Anmeldeinformationen (empfohlen).
- Nutzen Sie einen bestimmten Anbieter bzw. eine Anbieterkette von Anmeldeinformationen (oder erstellen Sie Ihren eigenen).
- Geben Sie die temporären Anmeldeinformationen selbst im Code an.

Verwenden der standardmäßigen Anbieterkette von Anmeldeinformationen

Wenn Sie einen neuen Service-Client initialisieren, ohne Argumente anzugeben, AWS SDK for Java versucht die , temporäre Anmeldeinformationen mithilfe der von der StandardklasseAWSCredentialsProviderChain implementierten Standard-Anbieterkette für Anmeldeinformationen zu finden. Die standardmäßige Anbieterkette von Anmeldeinformationen sucht in dieser Reihenfolge nach Anmeldeinformationen:

1. Umgebungsvariablen –`AWS_ACCESS_KEY_IDAWS_SECRET_ACCESS_KEY`, und `AWS_SESSION_TOKEN`. Die AWS SDK for Java verwendet die -EnvironmentVariableCredentialsProviderKlasse, um diese Anmeldeinformationen zu laden.
2. Java-Systemeigenschaften –`aws.accessKeyIdaws.secretKey`, und `aws.sessionToken`. Die AWS SDK for Java verwendet die SystemPropertiesCredentialsProvider, um diese Anmeldeinformationen zu laden.
3. Web-Identitätstoken-Anmeldeinformationen aus der Umgebung oder dem Container.
4. Die standardmäßige Datei mit den Anmeldeinformationsprofilen – befindet sich in der Regel unter `~/.aws/credentials` (der Standort kann je nach Plattform variieren) und wird von vielen der - AWS SDKs und von der gemeinsam genutzt AWS CLI. Die AWS SDK for Java verwendet die ProfileCredentialsProvider, um diese Anmeldeinformationen zu laden.

   Sie können eine Anmeldeinformationsdatei mit dem `aws configure` Befehl erstellen, der von der bereitgestellt wird AWS CLI, oder Sie können sie erstellen, indem Sie die Datei mit einem Texteditor bearbeiten. Informationen zum Dateiformat für Anmeldeinformationen finden Sie unter AWS Dateiformat für Anmeldeinformationen.
5. Amazon-ECS-Container-Anmeldeinformationen – werden aus Amazon ECS geladen, wenn die Umgebungsvariable festgelegt `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` ist. Die AWS SDK for Java verwendet die ContainerCredentialsProvider, um diese Anmeldeinformationen zu laden. Sie können die IP-Adresse für diesen Wert angeben.
6. Instance-Profilanmeldeinformationen – werden auf EC2-Instances verwendet und über den Amazon EC2 Metadatenservice bereitgestellt. Die AWS SDK for Java verwendet die

InstanceProfileCredentialsProvider, um diese Anmeldeinformationen zu laden. Sie können die IP-Adresse für diesen Wert angeben.

> **Note**
>
> Instance-Profil-Anmeldeinformationen werden nur verwendet, wenn `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` nicht gesetzt ist. Weitere Informationen finden Sie unter EC2ContainerCredentialsProviderWrapper.

Temporäre Anmeldeinformationen festlegen

Um AWS temporäre Anmeldeinformationen verwenden zu können, müssen sie an mindestens einem der vorherigen Speicherorte festgelegt werden. Weitere Informationen über das Festlegen von Anmeldeinformationen finden Sie in den folgenden Themen:

- Informationen zum Angeben von Anmeldeinformationen in der Umgebung oder in der Datei mit den Standard-Anmeldeinformationsprofilen finden Sie unter the section called “Verwenden temporäre Anmeldeinformationen” .
- Informationen über das Festlegen von Java-Systemeigenschaften finden Sie in der System Properties-Anleitung auf der offiziellen Java Tutorials-Website.
- Informationen zum Einrichten und Verwenden von Instance-Profilanmeldeinformationen mit Ihren EC2-Instances finden Sie unter Verwenden von IAM-Rollen zum Gewähren von Zugriff auf AWS Ressourcen auf Amazon EC2.

Einrichten eines alternativen Anmeldeinformationsprofils

Der AWS SDK for Java verwendet standardmäßig das Standardprofil, aber es gibt Möglichkeiten, anzupassen, welches Profil aus der Anmeldeinformationsdatei stammt.

Sie können die Umgebungsvariable AWS Profil verwenden, um das vom SDK geladene Profil zu ändern.

Auf Linux, macOS oder Unix würden Sie beispielsweise den folgenden Befehl ausführen, um das Profil in myProfile zu ändern.

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

Verwenden Sie unter Windows folgende Variante:

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

Das Festlegen der `AWS_PROFILE` Umgebungsvariablen wirkt sich auf das Laden von Anmeldeinformationen für alle offiziell unterstützten AWS SDKs und Tools (einschließlich AWS CLI und AWS Tools for Windows PowerShell) aus. Um nur das Profil für eine Java-Anwendung zu ändern, können Sie `aws.profile` stattdessen die -Systemeigenschaft verwenden.

> **Note**
>
> Die Umgebungsvariable hat Vorrang vor der Systemeigenschaft.

Festlegen eines Speicherorts für alternative Anmeldeinformationen

Der AWS SDK for Java lädt AWS temporäre Anmeldeinformationen automatisch aus dem Dateispeicherort der Standardanmeldeinformationen. Sie können jedoch auch den Speicherort angeben, indem Sie die Umgebungsvariable `AWS_CREDENTIAL_PROFILES_FILE` auf den vollständigen Pfad zur Anmeldeinformationsdatei setzen.

Sie können diese Funktion verwenden, um den Speicherort vorübergehend zu ändern, an dem nach Ihrer Anmeldeinformationsdatei AWS SDK for Java sucht (z. B. indem Sie diese Variable über die Befehlszeile festlegen). Alternativ können Sie die Umgebungsvariable in Ihrer Benutzer- oder Systemumgebung setzen, um sie für den Benutzer oder systemweit zu ändern.

So überschreiben Sie den Standardspeicherort der Anmeldeinformationsdatei

- Legen Sie die `AWS_CREDENTIAL_PROFILES_FILE` Umgebungsvariable auf den Speicherort Ihrer AWS Anmeldeinformationsdatei fest.
  - Verwenden Sie unter Linux, macOS oder Unix:

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

  - Verwenden Sie unter Windows:

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

`Credentials` Dateiformat

Wenn Sie die Anweisungen in der grundlegenden Einrichtung dieses Handbuchs befolgen, sollte Ihre Anmeldeinformationsdatei das folgende grundlegende Format haben.

```
[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>
```

Der Profilname wird in eckigen Klammern angegeben (zum Beispiel `[default]`), gefolgt von den konfigurierbaren Feldern in diesem Profil als Schlüssel-Wert-Paare. Sie können mehrere Profile in Ihrer -`credentials`Datei haben, die mit hinzugefügt oder bearbeitet werden können, `aws configure --profile` *`PROFILE_NAME`* um das zu konfigurierende Profil auszuwählen.

Sie können zusätzliche Felder angeben, z. B. `metadata_service_timeout`und `metadata_service_num_attempts`. Diese können nicht mit der CLI konfiguriert werden. Sie müssen die Datei manuell bearbeiten, wenn Sie sie verwenden möchten. Weitere Informationen zur Konfigurationsdatei und ihren verfügbaren Feldern finden Sie unter Konfigurieren der AWS Command Line Interface im AWS Command Line Interface -Benutzerhandbuch.

Laden von Anmeldeinformationen

Nachdem Sie temporäre Anmeldeinformationen festgelegt haben, lädt das SDK diese mithilfe der standardmäßigen Anbieterkette für Anmeldeinformationen.

Dazu instanziieren Sie einen - AWS-Service Client, ohne dem Builder explizit Anmeldeinformationen bereitzustellen, wie folgt.

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

Geben Sie einen Anbieter von Anmeldeinformationen oder eine Anbieterkette an

Sie können einen Anmeldeinformationsanbieter angeben, der sich von der standardmäßigen Anbieterkette von Anmeldeinformationen unterscheidet. Verwenden Sie dazu den Client-Generator.

Sie stellen eine Instance eines Anmeldeinformationsanbieters oder einer Anbieterkette für einen Client Builder bereit, der eine AWSCredentialsProvider Schnittstelle als Eingabe verwendet. Das folgende Beispiel zeigt konkret, wie Sie Anmeldeinformationen der Umgebung nutzen.

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

Eine vollständige Liste der von AWS SDK for Java bereitgestellten Anbieter von Anmeldeinformationen und Anbieterketten finden Sie unter Alle bekannten Implementieren von Klassen in AWSCredentialsProvider.

> **Note**
>
> Sie können diese Technik verwenden, um Anbieter von Anmeldeinformationen oder Anbieterketten bereitzustellen, die Sie mit Ihrem eigenen Anbieter von Anmeldeinformationen erstellen, der die `AWSCredentialsProvider` Schnittstelle implementiert, oder indem Sie die AWSCredentialsProviderChain Klasse unterteilen.

Explizite Angabe temporärer Anmeldeinformationen

Wenn die Standard-Anmeldeinformationen oder ein bestimmter oder benutzerdefinierter Anbieter oder eine Anbieterkette für Ihren Code nicht funktionieren, können Sie explizit angegebene Anmeldeinformationen festlegen. Wenn Sie temporäre Anmeldeinformationen mit abgerufen haben AWS STS, verwenden Sie diese Methode, um die Anmeldeinformationen für den AWS Zugriff anzugeben.

1. Instanziieren Sie die BasicSessionCredentials Klasse und stellen Sie ihr den AWS Zugriffsschlüssel, den AWS geheimen Schlüssel und das AWS Sitzungstoken zur Verfügung, das das SDK für die Verbindung verwendet.
2. Erstellen Sie ein AWSStaticCredentialsProvider mit dem -`AWSCredentials`Objekt.

3. Konfigurieren Sie den Client-Generator mit dem `AWSStaticCredentialsProvider` und erstellen Sie den Client.

Im Folgenden wird ein Beispiel gezeigt.

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

Weitere Infos

- Registrieren und AWS Erstellen eines IAM-Benutzers
- Einrichten von AWS Anmeldeinformationen und Regionen für die Entwicklung
- Verwenden von IAM-Rollen zum Gewähren des Zugriffs auf AWS Ressourcen in Amazon EC2

AWS-Region Auswahl

Regionen ermöglichen Ihnen den Zugriff auf AWS Services, die sich physisch in einem bestimmten geografischen Gebiet befinden. Dies ist nicht nur für die Redundanz nützlich, sondern sorgt auch dafür, dass Ihre Daten und Anwendungen in der Nähe Ihres Standorts sowie des Standorts Ihrer Benutzer ausgeführt werden.

Überprüfung der Serviceverfügbarkeit in einer Region

Um festzustellen, ob ein bestimmter in einer Region verfügbar AWS-Service ist, verwenden Sie die `-isServiceSupported`Methode für die Region, die Sie verwenden möchten.

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

Weitere Informationen über die Regionen, die Sie angeben können, und über die Nutzung des Endpunkt-Präfixes für den abzufragenden Service finden Sie in der Dokumentation der Regions-Klasse. Jedes Service-Endpunkt-Präfix wird in der Service-Schnittstelle definiert. Beispielsweise ist das DynamoDB Endpunktpräfix in der AmazonDynamoDB definiert.

Auswählen einer Region

Ab Version 1.4 des können AWS SDK for Java Sie einen Regionsnamen angeben und das SDK wählt automatisch einen geeigneten Endpunkt für Sie aus. Informationen darüber, wie Sie den Endpunkt selbst auswählen können, finden Sie unter Auswahl eines bestimmten Endpunkts.

Um explizit eine Region festzulegen, empfehlen wir, dass Sie die Regions-Aufzählung nutzen. Dabei handelt es sich um eine Aufzählung aller öffentlich verfügbaren Regionen. Mit dem folgenden Code können Sie einen Client mit einer Region aus der Aufzählung erstellen:

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

Wenn die Region, die Sie verwenden möchten, nicht in der `Regions`-Aufzählung enthalten ist, können Sie die Region mit einem String festlegen, der den Namen der Region enthält.

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

> **Note**
>
> Nachdem Sie einen Client mit dem Generator erstellt haben, ist er unveränderlich und die Region kann nicht mehr geändert werden. Wenn Sie mit mehreren AWS-Regionen für denselben Service arbeiten, sollten Sie mehrere Clients erstellen – einen pro Region.

Auswahl eines bestimmten Endpunkts

Jeder AWS Client kann so konfiguriert werden, dass er einen bestimmten Endpunkt innerhalb einer Region verwendet, indem die `withEndpointConfiguration` Methode beim Erstellen des Clients aufgerufen wird.

Um den Amazon S3 Client beispielsweise für die Verwendung der Region Europa (Irland) zu konfigurieren, verwenden Sie den folgenden Code.

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

```
            "eu-west-1"))
        .withCredentials(CREDENTIALS_PROVIDER)
        .build();
```

Die aktuelle Liste der Regionen und ihrer entsprechenden Endpunkte für alle AWS Services finden Sie unter Regionen und Endpunkte.

Automatisches Bestimmen der Region aus der Umgebung

> **Important**
>
> Dieser Abschnitt gilt nur, wenn ein Client Builder für den Zugriff auf AWS Services verwendet wird. AWS Clients, die mit dem Client-Konstruktor erstellt wurden, bestimmen nicht automatisch die Region aus der Umgebung und verwenden stattdessen die Standard-SDK-Region (USEast1).

Wenn Sie auf Amazon EC2 oder Lambda ausführen, sollten Sie Clients so konfigurieren, dass sie dieselbe Region verwenden, in der Ihr Code ausgeführt wird. So wird der Code von der Umgebung abgekoppelt, in der er läuft, wodurch die Bereitstellung Ihrer Anwendung in mehreren Regionen einfacher wird. Dies wiederum sorgt für weniger Latenz und mehr Redundanz.

Sie sollten Client-Generatoren verwenden, damit das SDK die Region, in der der Code ausgeführt wird, automatisch erkennt.

Rufen Sie die `defaultClient`-Methode des Client-Generators auf, um die Region aus der Umgebung mithilfe der Standard-Anbieterkette für Anmeldeinformationen/Regionen zu bestimmen.

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

Dies entspricht der Verwendung von `standard`, gefolgt von `build`.

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

Wenn Sie nicht explizit eine Region mit den `withRegion`-Methoden festlegen, nutzt das SDK die Standard-Anbieterkette für Regionen und versucht, die zu nutzende Region zu ermitteln.

Standard-Anbieterkette für Regionen

Folgende Regeln gelten für das Nachschlagen der Region:

1. Etwaige explizite, mit `withRegion` oder `setRegion` festgelegte Regionen direkt im Generator haben Vorrang vor allen anderen.
2. Die Umgebungsvariable `AWS_REGION` wird geprüft. Wenn sie festgelegt ist, wird die zugehörige Region zur Konfiguration des Clients verwendet.

   > **Note**
   >
   > Diese Umgebungsvariable wird vom Lambda Container festgelegt.

3. Das SDK überprüft die AWS freigegebene Konfigurationsdatei (normalerweise unter `~/.aws/config`). Ist die Eigenschaft region vorhanden, wird sie vom SDK verwendet.
   - Die Umgebungsvariable `AWS_CONFIG_FILE` kann verwendet werden, um den Speicherort der gemeinsam genutzten Konfigurationsdatei anzupassen.
   - Die `AWS_PROFILE` Umgebungsvariable oder die `aws.profile` Systemeigenschaft kann verwendet werden, um das vom SDK geladene Profil anzupassen.
4. Das SDK versucht, den Amazon EC2 Instance-Metadatenservice zu verwenden, um die Region der aktuell Amazon EC2 ausgeführten Instance zu bestimmen.
5. Hat das SDK zu diesem Zeitpunkt immer noch keine Region gefunden, schlägt die Erstellung des Clients mit einer Ausnahme fehl.

Bei der Entwicklung von AWS Anwendungen besteht ein gängiger Ansatz darin, die freigegebene Konfigurationsdatei (beschrieben unter Verwendung der standardmäßigen Anbieterkette für Anmeldeinformationen) zu verwenden, um die Region für die lokale Entwicklung festzulegen, und sich auf die standardmäßige Anbieterkette der Region zu verlassen, um die Region zu bestimmen, wenn sie auf der - AWS Infrastruktur ausgeführt wird. Dies vereinfacht die Client-Erstellung stark und sorgt dafür, dass Ihre Anwendung portabel bleibt.

Umgang mit Ausnahmen

Es ist wichtig zu verstehen, wie und wann Ausnahmen AWS SDK for Java auslöst, um mit dem SDK qualitativ hochwertige Anwendungen zu erstellen. In den folgenden Abschnitten werden die

verschiedenen Fälle von Ausnahmen beschrieben, die vom SDK ausgelöst werden, und wie sie korrekt verarbeitet werden.

Warum ungeprüfte Ausnahmen?

Der AWS SDK for Java verwendet Laufzeitausnahmen (oder nicht überprüfte Ausnahmen) anstelle von überprüften Ausnahmen aus folgenden Gründen:

- Entwickler erhalten genaue Kontrolle über die Fehler, auf die sie eingehen möchten. Sie werden aber nicht dazu gezwungen, auftretende Ausnahmen zu verarbeiten, für die sie sich nicht interessieren (was den Code übermäßig aufblähen würde).
- Skalierbarkeitsprobleme durch geprüfte Ausnahmen in großen Anwendungen werden verhindert.

Im Allgemeinen eignen sich geprüfte Ausnahmen gut im kleinen Rahmen. Wenn Anwendungen wachsen und komplexer werden, können sie allerdings zu Problemen führen.

Weitere Informationen über die Verwendung von geprüften und ungeprüften Ausnahmen finden Sie unter:

- Nicht aktivierte Ausnahmen – Die Konflikte
- The Trouble with Checked Exceptions
- Java's checked exceptions were a mistake (and here's what I would like to do about it)

AmazonServiceException (und Unterklassen)

AmazonServiceException ist die häufigste Ausnahme, die bei der Verwendung der auftritt AWS SDK for Java. Diese Ausnahme stellt eine Fehlerantwort von einem dar AWS-Service. Wenn Sie beispielsweise versuchen, eine Instance zu beenden, Amazon EC2 die nicht vorhanden ist, gibt EC2 eine Fehlerantwort zurück und alle Details dieser Fehlerantwort werden in das `AmazonServiceException` ausgegebene aufgenommen. In einigen Fällen wird eine abgeleitete Klasse von `AmazonServiceException` ausgelöst. So erhalten Entwickler genaue Kontrolle über den Umgang mit Fehlerfällen in Catch-Blöcken.

Wenn Sie auf eine stoßen`AmazonServiceException`, wissen Sie, dass Ihre Anfrage erfolgreich an die gesendet wurde, AWS-Service aber nicht erfolgreich verarbeitet werden konnte. Dies kann an Fehlern in den Parametern der Anforderung oder an Problemen auf Seiten des Services liegen.

`AmazonServiceException` gibt Ihnen Informationen wie z. B.:

- zurückgegebener HTTP-Statuscode
- Zurückgegebener AWS Fehlercode
- detaillierte Fehlermeldung aus dem Service
- AWS Anforderungs-ID für die fehlgeschlagene Anforderung

`AmazonServiceException` enthält auch Informationen darüber, ob die fehlgeschlagene Anforderung der Fehler des Aufrufers (eine Anforderung mit unzulässigen Werten) oder der Fehler AWS-Service des (ein interner Servicefehler) war.

AmazonClientException

AmazonClientException gibt an, dass im Java-Clientcode ein Problem aufgetreten ist, entweder beim Versuch, eine Anfrage an zu senden, AWS oder beim Versuch, eine Antwort von zu analysieren AWS. Ein `AmazonClientException` ist im Allgemeinen schwerwiegender als ein `AmazonServiceException`und weist auf ein großes Problem hin, das den Client daran hindert, Service-Aufrufe an - AWS Services durchzuführen. Beispielsweise AWS SDK for Java löst ein aus, `AmazonClientException` wenn keine Netzwerkverbindung verfügbar ist, wenn Sie versuchen, eine -Operation auf einem der Clients aufzurufen.

Asynchrone Programmierung

Sie können entweder synchrone oder asynchrone Methoden verwenden, um -Operationen auf - AWS Services aufzurufen. Synchrone Methoden blockieren die Ausführung Ihres Threads, bis der Client eine Antwort vom Service erhält. Asynchrone Methoden kehren sofort zurück. So haben Sie die Gewissheit, dass die Kontrolle an den aufrufenden Thread zurückgegeben wird, ohne auf eine Antwort zu warten.

Da eine asynchrone Methode zurückmeldet, bevor eine Antwort verfügbar ist, benötigen Sie einen Weg, an die Antwort zu gelangen, sobald diese bereitsteht. Die AWS SDK for Java bietet zwei Möglichkeiten: Zukünftige Objekte und Rückrufmethoden.

Java-Futures

Asynchrone Methoden im AWS SDK for Java geben ein future-Objekt zurück, das die Ergebnisse der asynchronen Operation in der future enthält.

Rufen Sie die `Future isDone()`-Methode auf, um festzustellen, ob der Service bereits ein Antwortobjekt bereitgestellt hat. Wenn die Antwort bereit ist, können Sie das Antwortobjekt durch

Aufrufen der `Future` `get()`-Methode erhalten. Mit diesem Mechanismus können Sie regelmäßig eine Abfrage nach den Ergebnissen der asynchronen Operation durchführen, während die Anwendung an anderen Aufgaben arbeitet.

Hier ist ein Beispiel für eine asynchrone Operation, die eine - Lambda Funktion aufruft und ein empfängt`Future`, das ein InvokeResult Objekt enthalten kann. Das `InvokeResult`-Objekt wird erst abgerufen, sobald `isDone()` `true` ergibt.

```
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);
    }
}
```

Asynchrone Callbacks

Zusätzlich zur Verwendung des Java-`Future`Objekts zur Überwachung des Status asynchroner Anforderungen ermöglicht Ihnen das SDK auch die Implementierung einer Klasse, die die -AsyncHandlerSchnittstelle verwendet. `AsyncHandler` bietet zwei Methoden, die je nachdem aufgerufen werden, wie die Anforderung abgeschlossen wurde: on`Success` und on`Error`.

Der wichtigste Vorteil der Rückruf-Schnittstelle besteht darin, dass Sie das `Future`-Objekt nicht mehr regelmäßig abfragen müssen, um zu ermitteln, wann die Anforderung abgeschlossen wurde. Stattdessen kann Ihr Code sofort die nächste Aktivität beginnen und sich auf das SDK verlassen, das für den Aufruf Ihrer Handler zum richtigen Zeitpunkt sorgt.

```
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(".");
        }
    }
}
```

Bewährte Methoden

Ausführen eines Rückrufs

Ihre Implementierung des `AsyncHandler` wird in dem Threadpool ausgeführt, der dem asynchronen Client gehört. Der kurze, schnell ausgeführte Code sollte am besten in der `AsyncHandler`-Implementierung eingefügt werden. Der lange ausgeführte oder blockierende Code in Handler-Methoden kann zu Konkurrenzsituationen für den Threadpool, der vom asynchronen Client verwendet wird, und kann den Client daran hindern, Anforderungen auszuführen. Wenn Sie eine lang andauernde Aufgabe haben, die mit einem Rückruf gestartet werden soll, lassen Sie den Rückruf die Aufgabe in einem neuen Thread oder in einem Threadpool ausführen, der von Ihrer Anwendung verwaltet wird.

Threadpool-Konfiguration

Die asynchronen Clients in der AWS SDK for Java bieten einen Standard-Threadpool, der für die meisten Anwendungen funktionieren sollte. Sie können einen benutzerdefinierten implementieren ExecutorService und ihn an AWS SDK for Java asynchrone Clients übergeben, um mehr Kontrolle über die Verwaltung der Thread-Pools zu erhalten.

Sie können beispielsweise eine `ExecutorService` Implementierung bereitstellen, die einen benutzerdefinierten verwendet, ThreadFactory um zu steuern, wie Threads im Pool benannt werden, oder um zusätzliche Informationen zur Thread-Nutzung zu protokollieren.

Asynchroner Zugriff

Die TransferManager Klasse im SDK bietet asynchrone Unterstützung für die Arbeit mit Amazon S3. `TransferManager` verwaltet asynchrone Uploads und Downloads, bietet detaillierte Fortschrittsberichte zu Übertragungen und unterstützt Rückrufe für verschiedene Ereignisse.

Protokollieren von AWS SDK for Java Aufrufen

Die AWS SDK for Java ist mit Apache Commons Logging instrumentiert. Dabei handelt es sich um eine Abstraktionsebene, die die Verwendung eines beliebigen von mehreren Protokollierungssystemen zur Laufzeit ermöglicht.

Unterstützte Protokollierungssysteme sind u. a. das Java Logging Framework und Apache Log4j. In diesem Thema erhalten Sie Informationen zur Nutzung von Log4j. Sie können die Protokollierungsfunktionalität des SDKs ohne Änderungen am Code Ihrer Anwendung nutzen.

Weitere Informationen über Log4j finden Sie auf der Apache-Website.

> **Note**
>
> In diesem Thema geht es um Log4j 1.x. Log4j2 unterstützt Apache Commons Logging nicht direkt. Stattdessen wird ein Adapter bereitgestellt, der Protokollierungsaufrufe automatisch mithilfe der Apache Commons Logging-Schnittstelle an Log4j2 weiterleitet. Weitere Informationen finden Sie unter Commons Logging Bridge in der Log4j2-Dokumentation.

Herunterladen der Log4J-JAR

Zur Nutzung von Log4j mit dem SDK müssen Sie das Log4j-JAR von der Apache-Website herunterladen. Das SDK enthält das JAR nicht. Kopieren Sie die JAR-Datei an einen Speicherort, der in Ihrem Klassenpfad enthalten ist.

Log4j verwendet eine Konfigurationsdatei namens "log4j.properties". Beispiel-Konfigurationsdateien werden nachfolgend angezeigt. Kopieren Sie die Konfigurationsdatei in ein Verzeichnis in Ihrem Klassenpfad. Die Log4j JAR-Dateien und die Datei "log4j.properties" müssen nicht im selben Verzeichnis liegen.

In der Konfigurationsdatei "log4j.properties" sind Eigenschaften wie die Protokollierungsebene, das Ziel der Protokollierungsausgaben (z. B. an eine Datei oder an die Konsole) sowie das Ausgabeformat angegeben. Die Protokollierungsebene ist die Granularität der Ausgaben, die der Protokollierer erzeugt. Log4j unterstützt das Konzept mehrerer Hierarchien der Protokollierung. Die Protokollierungsebene wird für jede Hierarchie separat festgelegt. Die folgenden zwei Protokollierungshierarchien sind im AWS SDK for Java verfügbar:

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

Festlegen des Klassenpfads

Sowohl die Log4j JAR-Datei als auch die Datei "log4j.properties" müssen in Ihrem Klassenpfad liegen. Wenn Sie Apache Ant verwenden, legen Sie den Klassenpfad im `path`-Element der Ant-Datei fest. Das folgende Beispiel zeigt ein Pfadelement aus der Ant-Datei für das Amazon S3 Beispiel, das im SDK enthalten ist.

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

In der Eclipse-IDE können Sie den Klassenpfad festlegen, indem Sie das Menü öffnen und auf Projekt | Eigenschaften | Java Build-Pfad klicken.

Service-spezifische Fehler und Warnungen

Wir empfehlen, dass Sie die Protokollierungshierarchie "com.amazonaws" immer auf "WARN" gestellt lassen. So entgehen Ihnen keine wichtigen Meldungen aus den Client-Bibliotheken. Wenn der Amazon S3 Client beispielsweise feststellt, dass Ihre Anwendung ein nicht ordnungsgemäß geschlossen hat `InputStream` und Ressourcen preisgeben könnte, meldet der S3-Client es über eine Warnmeldung an die Protokolle. Dadurch wird auch sichergestellt, dass Nachrichten protokolliert werden, wenn der Client Schwierigkeiten bei der Verarbeitung von Anforderungen oder Antworten hat.

In der folgenden "log4j.properties"-Datei ist der `rootLogger` auf WARN gesetzt. Dies hat zur Folge, dass Warn- und Fehlermeldungen von allen Protokollierern in der Hierarchie "com.amazonaws" enthalten sind. Alternativ können Sie ausdrücklich den com.amazonaws-Protokollierer auf WARN stellen.

```
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
```

Protokollierung von Anforderungs-/Antwortübersichten

Jede Anfrage an einen AWS-Service generiert eine eindeutige AWS Anfrage-ID, die nützlich ist, wenn ein Problem damit auftritt, wie ein eine Anfrage AWS-Service verarbeitet. AWS Anfrage-IDs sind programmgesteuert über Ausnahmeobjekte im SDK für fehlgeschlagene Service-Aufrufe zugänglich und können auch über die DEBUG-Protokollebene im Logger „com.amazonaws.request“ gemeldet werden.

Die folgende Datei log4j.properties ermöglicht eine Zusammenfassung der Anforderungen und Antworten, einschließlich der AWS Anforderungs-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
```

Hier finden Sie ein Beispiel für die Protokollausgabe.

```
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
```

Verbose-Protokollierung des Netzwerkverkehrs

In einigen Fällen kann es nützlich sein, die genauen Anfragen und Antworten zu sehen, die AWS SDK for Java sendet und empfängt. Sie sollten diese Protokollierung nicht in Produktionssystemen aktivieren, da das Schreiben großer Anfragen (z. B. einer Datei, die in hochgeladen wird Amazon S3) oder Antworten eine Anwendung erheblich verlangsamen kann. Wenn Sie wirklich Zugriff auf diese Informationen benötigen, können Sie sie vorübergehend über den Apache- HttpClient 4-Logger

aktivieren. Durch Aktivieren der DEBUG-Ebene für den `org.apache.http.wire`-Protokollierer wird die Protokollierung für sämtliche Anforderungs- und Antwortdaten aktiviert.

Die folgende log4j.properties-Datei aktiviert die vollständige Protokollierung in Apache HttpClient 4 und sollte nur vorübergehend aktiviert werden, da sie erhebliche Auswirkungen auf die Leistung Ihrer Anwendung haben kann.

```
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
```

Protokollieren von Latenz-Metriken

Wenn Sie bei der Behandlung Metriken anzeigen möchten, z. B. welcher Prozess die meiste Zeit beansprucht oder ob die Latenz auf der Server- oder Client-Seite größer ist, kann der Latenz-Protokollierer hilfreich sein. Stellen Sie den `com.amazonaws.latency`-Protokollierer zur Aktivierung auf DEBUG.

> **Note**
>
> Dieser Protokollierer ist nur verfügbar, wenn SDK-Metriken aktiviert sind. Weitere Informationen zum SDK-Metrikpaket finden Sie unter Aktivieren von Metriken für die AWS SDK for Java.

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

Hier finden Sie ein Beispiel für die Protokollausgabe.

```
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]
```

Client-Konfiguration

Mit der AWS SDK for Java können Sie die Standard-Client-Konfiguration ändern, was hilfreich ist, wenn Sie:

- Herstellen einer Internetverbindung über einen Proxy
- Ändern von HTTP-Transport-Einstellungen, z. B. Verbindungstimeout und wiederholte Anforderungsversuche
- Angabe von TCP-Socketpuffer-Größenhinweisen

Proxy-Konfiguration

Beim Erstellen eines Clientobjekts können Sie ein optionales ClientConfiguration Objekt übergeben, um die Konfiguration des Clients anzupassen.

Wenn Sie sich über einen Proxy-Server mit dem Internet verbinden, sollten Sie über das `ClientConfiguration`-Objekt die Einstellungen des Proxy-Servers konfigurieren (Proxy-Host, -Port und Benutzername/Passwort).

HTTP-Transport-Konfiguration

Sie können mehrere HTTP-Transportoptionen konfigurieren, indem Sie das -ClientConfigurationObjekt verwenden. Neue Optionen werden gelegentlich hinzugefügt. Eine vollständige Liste der Optionen, die Sie abrufen oder festlegen können, finden Sie in der AWS SDK for Java -API-Referenz.

> **Note**
>
> Jeder konfigurierbare Wert hat einen von einer Konstanten definierten Standardwert. Eine Liste der konstanten Werte für finden Sie unter `ClientConfiguration`Konstante

Feldwerte in der AWS SDK for Java API-Referenz zu . https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/constant-values.html

Maximale Anzahl der Verbindungen

Sie können die maximal zulässige Anzahl offener HTTP-Verbindungen mithilfe der ClientConfigurationsetMaxConnections Methode festlegen.

> **Important**
>
> Legen Sie die maximalen Verbindungen auf die Anzahl der gleichzeitigen Transaktionen fest, um Verbindungskonflikte und eine schlechte Leistung zu vermeiden. Informationen zum standardmäßigen Wert für maximale Verbindungen finden Sie unter Konstante Feldwerte in der AWS SDK for Java API-Referenz zu .

Timeouts und Fehlerbehandlung

Sie können Optionen im Zusammenhang mit Timeouts und der Fehlerbehandlung für HTTP-Verbindungen festlegen.

- Verbindungstimeout

  Das Verbindungstimeout ist die Zeit (in Millisekunden), die die HTTP-Verbindung wartet, um eine Verbindung herzustellen. Der Standardwert beträgt 10 000 ms.

  Um diesen Wert selbst festzulegen, verwenden Sie die Methode ClientConfiguration.setConnectionTimeout

- Connection Time to Live (TTL, Gültigkeitsdauer der Verbindung)

  Standardmäßig versucht das SDK, HTTP-Verbindungen so lange wie möglich wiederzuverwenden. In Fehlersituationen, bei denen eine Verbindung zu einem Server aufgebaut wird, der außer Betrieb genommen wird, sorgt eine endliche TTL für eine schnellere Anwendungswiederherstellung. Wird beispielsweise eine 15-Minuten-TTL eingestellt, ist dadurch sichergestellt, dass auch dann, wenn Sie eine Verbindung zu einem Server aufgebaut haben, bei dem Probleme auftreten, innerhalb von 15 Minuten eine Verbindung zu einem neuen Server hergestellt wird.

Um die HTTP-Verbindungs-TTL festzulegen, verwenden Sie die Methode ClientConfiguration.setConnectionTTL.

- Maximale Anzahl der erneuten Versuche

  Die standardmäßige maximale Wiederholungsanzahl für Fehler bei wiederholbaren Aktionen ist 3. Sie können einen anderen Wert festlegen, indem Sie die Methode ClientConfiguration.setMaxErrorRetry verwenden.

Lokale Adresse

Um die lokale Adresse festzulegen, an die der HTTP-Client gebunden wird, verwenden Sie ClientConfiguration.setLocalAddress

TCP-Socketpuffer-Größenhinweise

Fortgeschrittene Benutzer, die TCP-Parameter auf niedriger Ebene optimieren möchten, können zusätzlich TCP-Puffergrößenhinweise über das -ClientConfigurationObjekt festlegen. Die meisten Benutzer müssen diese Einstellungen nie anpassen, sie stehen aber für fortgeschrittene Benutzer bereit.

Die optimalen TCP-Puffergrößen für eine Anwendung hängen stark von der Konfiguration und den Fähigkeiten des Netzwerks und des Betriebssystems ab. Beispielsweise bieten die meisten modernen Betriebssysteme eine Logik zur automatischen Anpassung der TCP-Puffergrößen. Dies kann sich weitreichend auf die Leistung von TCP-Verbindungen auswirken, wenn diese lang genug geöffnet bleiben, damit die automatische Anpassung die Puffergrößen optimieren kann.

Große Puffer (z. B. 2 MB) ermöglichen dem Betriebssystem, mehr Daten im Arbeitsspeicher zu puffern, ohne dass der Remote-Server die Informationen bestätigen muss. Sie sind also besonders nützlich, wenn das Netzwerk eine hohe Latenz aufweist.

Dies ist nur ein Hinweis, an den sich das Betriebssystem nicht halten muss. Bei Verwendung dieser Option sollten Benutzer immer die beim Betriebssystem konfigurierten Grenz- und Standardwerte überprüfen. In den meisten Betriebssystemen ist eine maximale TCP-Puffergröße konfiguriert, die nicht überschritten wird, es sei denn, Sie heben die maximale TCP-Puffergröße explizit an.

Für die Konfiguration von TCP-Puffergrößen und betriebssystemspezifischen TCP-Einstellungen stehen viele Ressourcen zur Verfügung, wie z. B.:

- Host Tuning

Zugriffskontrollrichtlinien

AWS Mit -Zugriffskontrollrichtlinien können Sie differenzierte Zugriffskontrollen für Ihre - AWS Ressourcen festlegen. Eine Zugriffsrichtlinie besteht aus einer Reihe von Anweisungen, die folgende Form annehmen:

Konto A darf Aktion B auf Ressource C ausführen, wenn Bedingung D gilt.

Wobei gilt:

- A ist der Prinzipal – das AWS-Konto , das eine Anforderung für den Zugriff auf oder die Änderung einer Ihrer AWS Ressourcen stellt.
- B ist die Aktion – Die Art und Weise, wie auf Ihre AWS Ressource zugegriffen oder diese geändert wird, z. B. das Senden einer Nachricht an eine - Amazon SQS Warteschlange oder das Speichern eines Objekts in einem - Amazon S3 Bucket.
- C ist die Ressource – Die AWS Entität, auf die der Prinzipal zugreifen möchte, z. B. eine Amazon SQS -Warteschlange oder ein in gespeichertes Objekt Amazon S3.
- D ist ein Satz von Bedingungen – Die optionalen Einschränkungen, die angeben, wann der Zugriff für den Prinzipal auf Ihre Ressource zugelassen oder verweigert werden soll. Viele ausdrucksstarke Bedingungen sind verfügbar, einige speziell für jeden Service. Beispielsweise können Sie mit Datumsbedingungen den Zugriff auf Ressourcen nur nach oder vor einem bestimmten Zeitpunkt zulassen.

Amazon S3 Beispiel

Das folgende Beispiel zeigt eine Richtlinie, die es jedem Benutzer ermöglicht, alle Objekte in einem Bucket zu lesen, aber den Zugriff auf das Hochladen von Objekten in diesen Bucket auf zwei bestimmte AWS-Konto(zusätzlich zum Konto des Bucket-Eigentümers) beschränkt.

```
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 Beispiel

Eine häufige Verwendung von Richtlinien besteht darin, eine - Amazon SQS Warteschlange zum Empfangen von Nachrichten von einem Amazon SNS-Thema zu autorisieren.

```
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));
```

Amazon SNS-Beispiel

Einige -Services bieten zusätzliche Bedingungen, die in -Richtlinien verwendet werden können. Amazon SNS bietet Bedingungen zum Zulassen oder Verweigern von Abonnements für SNS-Themen basierend auf dem Protokoll (z. B. E-Mail, HTTP, HTTPS Amazon SQS) und Endpunkt (z. B. E-Mail-Adresse, URL, Amazon SQS ARN) der Anforderung zum Abonnieren eines Themas.

```
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()));
```

Festlegen des JVM-TTL-Werts für DNS-Name-Lookups

Die Java Virtual Machine (JVM) speichert DNS-Namensauflösungen zwischen. Wenn die JVM einen Hostnamen in eine IP-Adresse auflöst, speichert sie die IP-Adresse für einen bestimmten Zeitraum zwischen, der als time-to-live (TTL) bezeichnet wird.

Da AWS Ressourcen DNS-Namenseinträge verwenden, die sich gelegentlich ändern, empfehlen wir Ihnen, Ihre JVM mit einem TTL-Wert von nicht mehr als 60 Sekunden zu konfigurieren. Auf diese Weise wird bei Änderung der IP-Adresse einer Ressource sichergestellt, dass Ihre Anwendung die neue IP-Adresse der Ressource durch erneute Abfrage des DNS abrufen und nutzen kann.

Bei einigen Java-Konfigurationen ist die JVM-Standard-TTL so festgelegt, dass DNS-Einträge nie aktualisiert werden, bis die JVM neu gestartet wird. Wenn sich also die IP-Adresse für eine - AWS Ressource ändert, während Ihre Anwendung noch ausgeführt wird, kann sie diese Ressource erst verwenden, wenn Sie die JVM manuell neu starten und die zwischengespeicherten IP-Informationen aktualisiert werden. In diesem Fall ist es wichtig, die TTL der JVM so einzustellen, dass sie die zwischengespeicherten IP-Daten von Zeit zu Zeit aktualisiert.

> **Note**
>
> Die Standard-TTL kann je nach Version Ihrer JVM und abhängig davon, ob ein Sicherheits-Manager installiert ist, unterschiedlich sein. Viele JVMs bieten eine Standard-TTL von weniger als 60 Sekunden. Wenn Sie eine solche JVM und keinen Sicherheits-Manager nutzen, können Sie den Rest dieses Themas ignorieren.

So legen Sie die JVM-TTL fest

Sie können die TTL der JVM ändern, indem Sie den Eigenschaftswert networkaddress.cache.ttl festlegen. Nutzen Sie dazu eine der folgenden Methoden je nach Ihrem Bedarf:

- global für alle Anwendungen, die die JVM nutzen. Legen Sie `networkaddress.cache.ttl` in der `$JAVA_HOME/jre/lib/security/java.security` Datei für Java 8 oder in der `$JAVA_HOME/conf/security/java.security` Datei für Java 11 oder höher fest:

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

- nur für Ihre Anwendung, legen Sie `networkaddress.cache.ttl` im Initialisierungscode Ihrer Anwendung fest:

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

Aktivieren von Metriken für die AWS SDK for Java

kann Metriken für die Visualisierung und Überwachung mit Amazon CloudWatch AWS SDK for Java generieren, die Folgendes messen:

- Die Leistung Ihrer Anwendung beim Zugriff auf AWS
- die Leistung Ihrer JVMs bei Verwendung mit AWS
- Details der Laufzeitumgebung wie z. B. Heap-Speicher, Anzahl der Threads und geöffneter Datei-Deskriptoren

So aktivieren Sie die Generierung von Java-SDK-Metriken

Sie müssen die folgende Maven-Abhängigkeit hinzufügen, damit das SDK Metriken an senden kann 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>
```

*Ersetzen Sie die Versionsnummer durch die neueste Version des SDK, die unter Maven Central verfügbar ist.

AWS SDK for Java -Metriken sind standardmäßig deaktiviert. Um es für Ihre lokale Entwicklungsumgebung zu aktivieren, fügen Sie eine Systemeigenschaft hinzu, die beim Start der JVM auf Ihre Datei mit AWS Sicherheitsanmeldeinformationen verweist. Beispielsweise:

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

Sie müssen den Pfad zu Ihrer Anmeldeinformationsdatei angeben, damit das SDK die gesammelten Datenpunkte CloudWatch zur späteren Analyse in hochladen kann.

> **Note**
>
> Wenn Sie über den Amazon EC2 Instance-Metadatenservice von einer Amazon EC2 Instance AWS aus auf zugreifen, müssen Sie keine Datei mit Anmeldeinformationen angeben. In diesem Fall ist nur Folgendes anzugeben:
>
> ```
> -Dcom.amazonaws.sdk.enableDefaultMetrics
> ```

Alle von erfassten Metriken AWS SDK for Java befinden sich unter dem Namespace AWSSDK/Java und werden in die CloudWatch Standardregion (us-east-1) hochgeladen. Wenn Sie die Region ändern möchten, geben Sie sie mit dem Attribut `cloudwatchRegion` in der Systemeigenschaft an. Um beispielsweise die CloudWatch Region auf us-east-1 festzulegen, verwenden Sie:

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

Sobald Sie das Feature aktiviert haben, werden jedes Mal, wenn eine Serviceanfrage an AWS von der erfolgt AWS SDK for Java, Metrikdatenpunkte generiert, zur statistischen Zusammenfassung in die Warteschlange gestellt und CloudWatch etwa einmal pro Minute asynchron in hochgeladen. Sobald Metriken hochgeladen wurden, können Sie sie mithilfe der visualisieren AWS Management

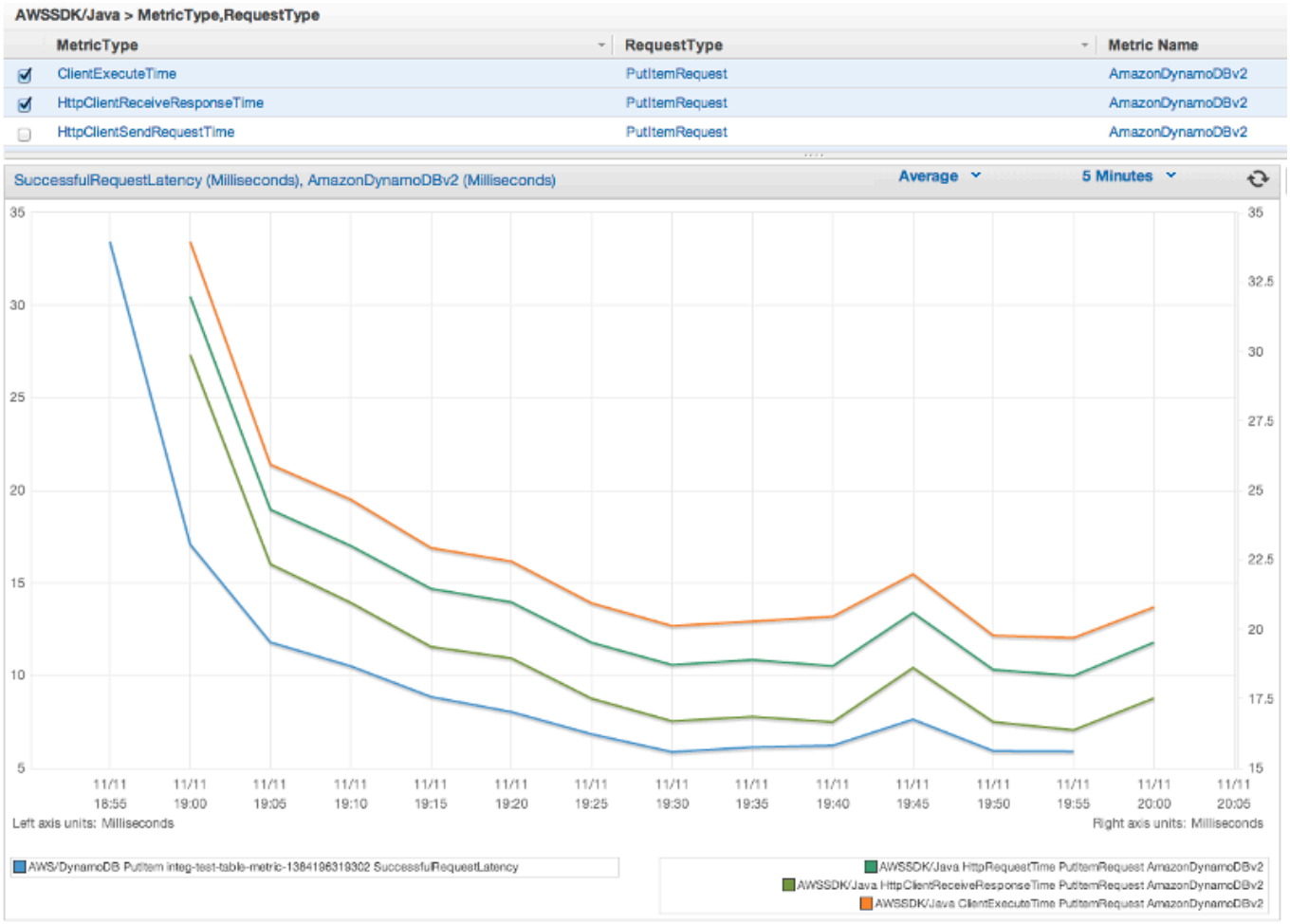
Console und Alarme für potenzielle Probleme wie Speicherlecks, Dateideskriptorlecks usw. einrichten.

Verfügbare Arten von Metriken

Die Standardmetriken werden in drei Hauptkategorien unterteilt:

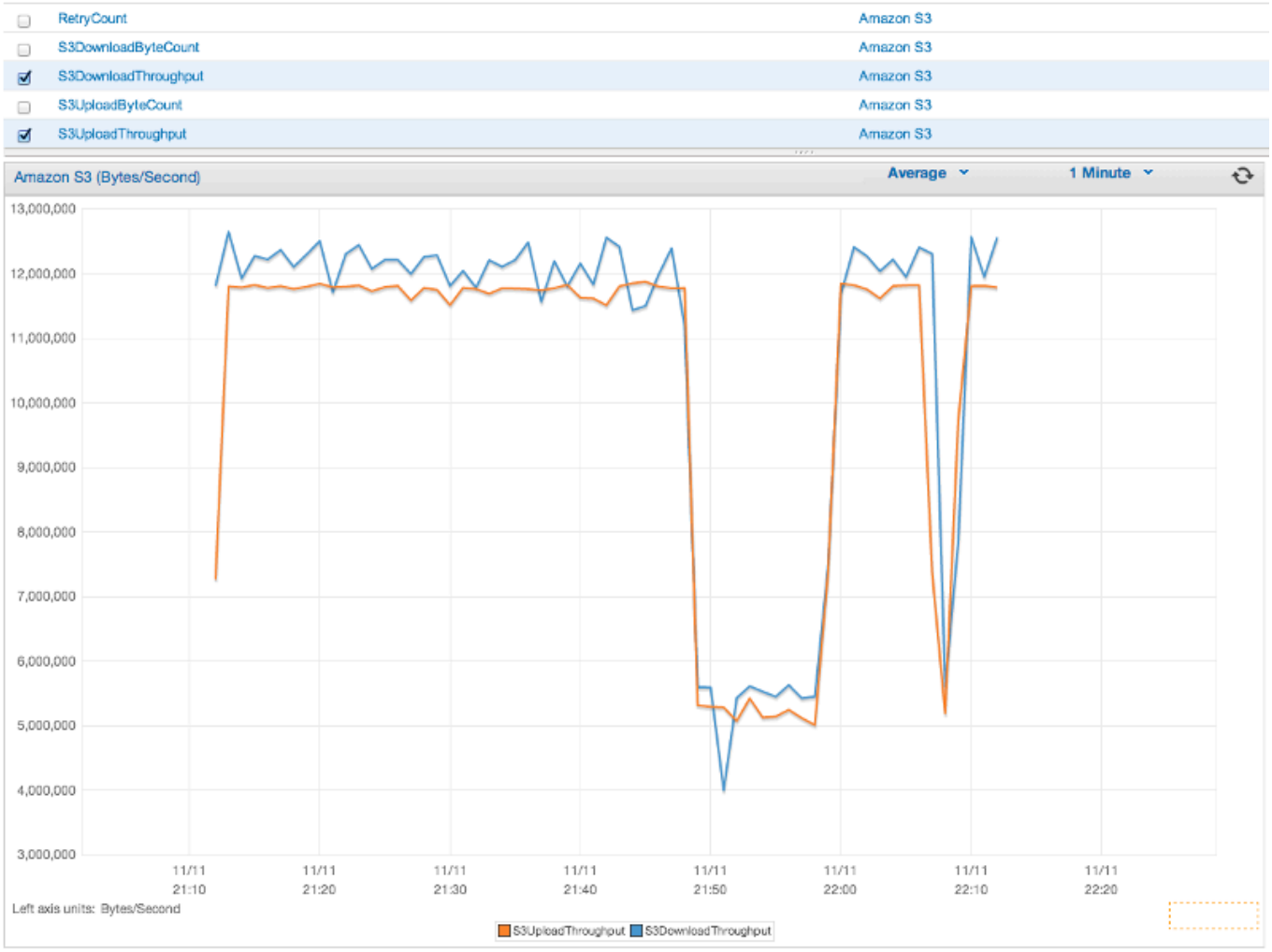
AWS Anforderungsmetriken

- Deckt Bereiche wie die Latenz der HTTP-Anforderung/-Antwort, die Anzahl der Anfragen, Ausnahmen und Wiederholungen ab.



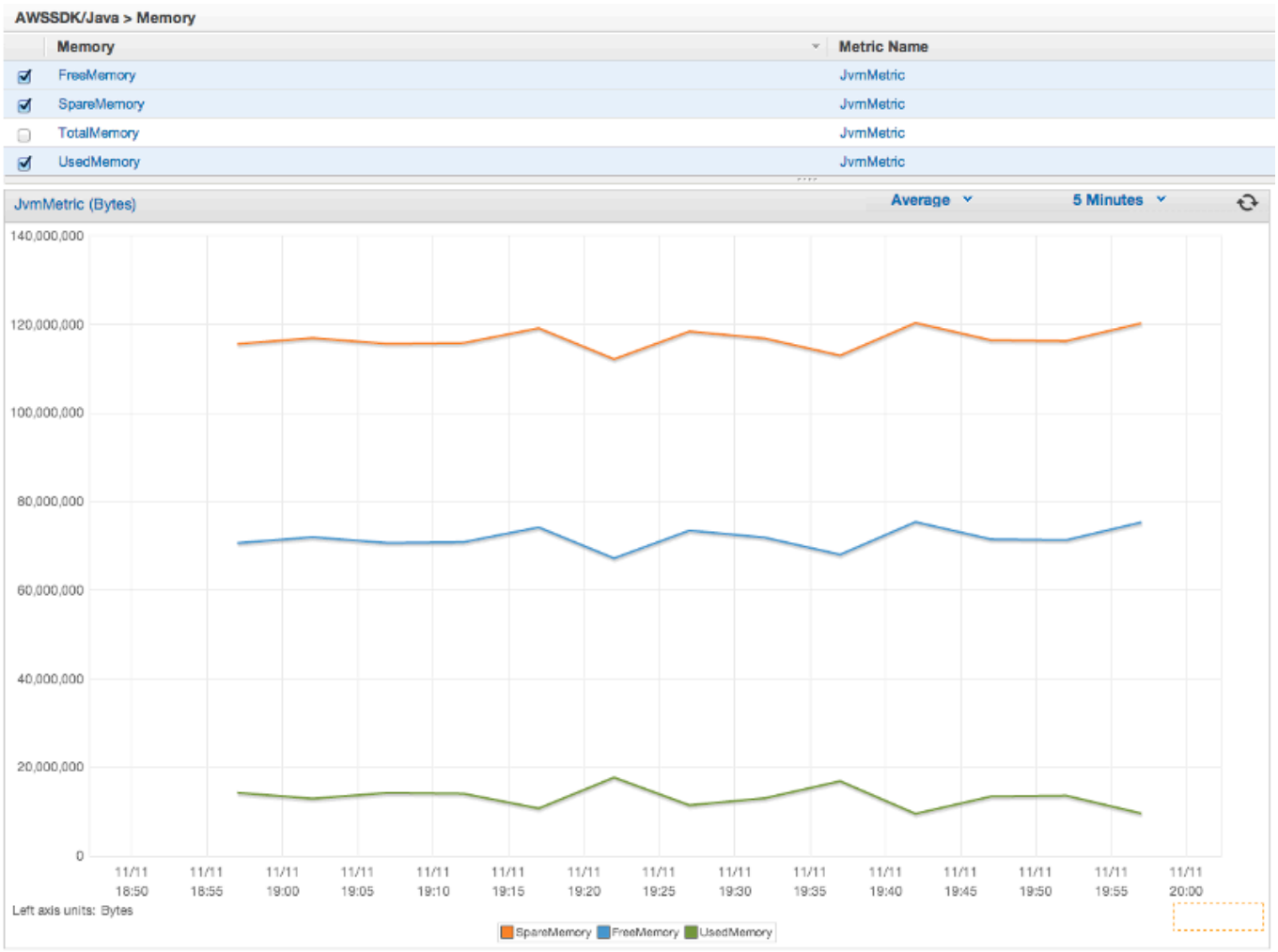
AWS-Service Metriken

- Fügen Sie AWS-Service-spezifische Daten wie Durchsatz und Byteanzahl für S3-Uploads und -Downloads ein.



Maschinenmetriken

- Enthalten die Laufzeitumgebung, darunter Heap-Speicher, Anzahl der Threads und geöffneter Datei-Deskriptoren.



Wenn Sie Maschinenmetriken ausschließen möchten, fügen Sie `excludeMachineMetrics` in die Systemeigenschaft ein:

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

Weitere Informationen

- In der amazonaws/metrics-Paketübersicht finden Sie eine vollständige Liste der vordefinierten Standardmetrik-Typen.
- Weitere Informationen zum Arbeiten mit CloudWatch mithilfe der finden Sie AWS SDK for Java unter CloudWatch Beispiele für die Verwendung der AWS SDK for Java.

- Weitere Informationen zur Leistungsoptimierung finden Sie im Blogbeitrag Optimieren der AWS SDK for Java zur Verbesserung der Ausfallsicherheit.

AWS SDK for Java-Codebeispiele

Dieser Abschnitt enthält Tutorials und Beispiele für die Verwendung derAWS SDK for Java v1 zum Programmieren vonAWS Diensten.

Den Quellcode für diese und andere Beispiele finden Sie im Repository fürAWS Dokumentationscodebeispiele unterGitHub.

Um ein neues Codebeispiel für das AWS-Dokumentationsteam vorzuschlagen, dessen Erstellung es erwägen soll, erstellen Sie eine neue Anfrage. Das Team möchte Codebeispiele erstellen, die breitere Szenarien und Anwendungsfälle abdecken, im Vergleich zu einfachen Codeausschnitten, die nur einzelne API-Aufrufe abdecken. Anweisungen finden Sie in den Richtlinien für Beiträge im Repositorium für Codebeispiele aufGitHub..

AWS SDK for Java2.x

Im Jahr 2018AWS veröffentlichte die AWS SDK for Java 2.x. Dieses Handbuch enthält Anweisungen zur Verwendung des neuesten Java-SDK sowie Beispielcode.

> **Note**
>
> Unter Zusätzliche Dokumentation und Ressourcen finden Sie weitere Beispiele und zusätzliche Ressourcen, die AWS SDK for Java-Entwicklern zur Verfügung stehen!

Verwendung der CloudWatch BeispieleAWS SDK for Java

Dieser Abschnitt bietet Beispiele für die Programmierung von CloudWatch mithilfe des AWS SDK for Java.

Amazon CloudWatch überwacht IhreAmazon Web Services(AWS) -Ressourcen und die Anwendungen, die Sie auf ausführenAWSin Echtzeit Sie können verwenden CloudWatch um Metriken zu erfassen und nachzuverfolgen, die Variablen sind, die Sie für Ihre Ressourcen und Anwendungen messen können. CloudWatch -Alarme senden Benachrichtigungen oder nehmen automatisch Änderungen an den Ressourcen vor, die Sie gemäß den von Ihnen festgelegten Regeln überwachen.

Weitere Informationen zu CloudWatch finden Sie im Amazon CloudWatch Benutzerhandbuch.

> **Note**
>
> Die Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der komplette Beispiel-Code steht auf GitHub bereit. Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

Themen

Abrufen von Metriken aus CloudWatch

Auflisten von Metriken

Zur Auflistung CloudWatch Metriken, erstellen Sie eineListMetricsRequestund rufen Sie den AmazonCloudWatchClient's an`listMetrics`-Methode. Sie können `ListMetricsRequest` zum Filtern der zurückgegebenen Metriken nach Namespace, Metrikname oder Dimensionen verwenden.

> **Note**
>
> Eine Liste der Metriken und Dimensionen, die vonAWSDienste finden Sie im {https—docs-AWS-Amazon-com-AmazoncloudWatch-Latest-Monitoring-CW-Support-for-AWS-html} [Amazon CloudWatch Referenz für -Metriken und -Dimensionen]Amazon CloudWatch-Benutzerhandbuch.

Importe

```
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;
```

Code

```
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;
    }
}
```

Die Metriken werden in einem ListMetricsResult durch Aufrufen der `getMetrics`-Methode zurückgegeben. Eventuell werden die Ergebnisse seitenweise zurückgegeben. Um den nächsten Stapel Ergebnisse abzurufen, rufen Sie `setNextToken` beim Original-Anforderungsobjekt mit dem Rückgabewert der `getNextToken`-Methode des `ListMetricsResult`-Objekts auf. Übergeben Sie das geänderte Anforderungsobjekt dann an einen weiteren Aufruf von `listMetrics`.

Weitere Informationen

- ListMetricsimAmazon CloudWatch-API-Referenz.

Veröffentlichen benutzerdefinierter Metrikdaten

Eine Reihe vonAWSDienste veröffentlichenihre eigenen Metrikenin Namespaces beginnend mit“AWS „Sie können auch benutzerdefinierte Metrikdaten unter Verwendung Ihres eigenen Namespace veröffentlichen (solange dieser nicht beginnt“AWS „).

Veröffentlichen benutzerdefinierter Metrikdaten

Rufen Sie zum Veröffentlichen eigener Metrikdaten des AmazonCloudWatchClients`putMetricData`-Methode mit einerPutmetricDataRequestaus. Die `PutMetricDataRequest` sollte den benutzerdefinierten Namespace enthalten, der für die Daten verwendet werden soll. Außerdem werden Infos über den Datenpunkt selbst in einem MetricDatum-Objekt erwartet.

> **Note**
>
> Sie können keinen Namespace angeben, der mit“ beginntAWS „. Namespaces die anfangen mit“AWS „sind für die Verwendung vonAmazon Web Services-Produkte.

Importe

```
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;
```

Code

```
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);
```

Weitere Informationen

- benutzenAmazon CloudWatchMetrikenimAmazon CloudWatch-Benutzerhandbuch.
- AWSNamespacesimAmazon CloudWatch-Benutzerhandbuch.
- PutMetricDataimAmazon CloudWatchAPI-Referenz.

Arbeiten mit CloudWatch Alarme

Einrichten eines Alarms

So erstellen Sie einen Alarm basierend auf einer CloudWatch metric, rufen Sie die AmazonCloudWatchClient's`putMetricAlarm`-Methode mit einerPutMetricAlarmRequestgefüllt mit den Alarmbedingungen.

Importe

```
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;
```

Code

```
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);
```

Auflisten von Alarmen

So listen Sie die CloudWatch Alarme, die Sie erstellt haben, rufen Sie die AmazonCloudWatchClient's`describeAlarms`-Methode mit einer[DescribeAlarmsRequest]die Sie verwenden können, um Optionen für das Ergebnis festzulegen.

Importe

```
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;
```

Code

```
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;
    }
}
```

Die Liste von Alarmen erhalten Sie, indem Sie `getMetricAlarms` auf dem DescribeAlarmsResult aufrufen, das von `describeAlarms` zurückgegeben wird.

Eventuell werden die Ergebnisse seitenweise zurückgegeben. Um den nächsten Stapel Ergebnisse abzurufen, rufen Sie `setNextToken` beim Original-Anforderungsobjekt mit dem Rückgabewert der `getNextToken`-Methode des `DescribeAlarmsResult`-Objekts auf. Übergeben Sie das geänderte Anforderungsobjekt dann an einen weiteren Aufruf von `describeAlarms`.

> **Note**
>
> Sie können auch Alarme für eine bestimmte Metrik abrufen, indem Sie den AmazonCloudWatchClients verwenden`describeAlarmsForMetric`-Methode. Sie lässt sich ähnlich wie `describeAlarms` nutzen.

Löschen von Alarmen

Um zu löschen CloudWatch Alarme, rufen Sie den AmazonCloudWatchClient's`deleteAlarms`-Methode mit einerdeleteAlarmsRequestSie enthält einen oder mehrere Namen von Alarmen, die Sie löschen möchten.

Importe

```
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;
```

Code

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

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

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

Weitere Informationen

- ErstellenAmazon CloudWatchAlarmeimAmazon CloudWatch-Benutzerhandbuch
- PutMetricAlarmimAmazon CloudWatch-API-Referenz
- DescribeAlarmsimAmazon CloudWatch-API-Referenz
- DeleteAlarmsimAmazon CloudWatch-API-Referenz

Verwenden von Alarmaktionen in CloudWatch

benutzen CloudWatch Mithilfe von -Alarmaktionen können Sie Alarme erstellen, die automatisch Aktionen wie etwa das Anhalten, Beenden, Neustarten oder Wiederherstellen ausführenAmazon EC2Instanzen.

> **Note**
>
> Mit der -Methode von PutMetricAlarmRequest`setAlarmActions` können Sie beim Erstellen eines Alarms Alarmaktionen hinzufügen.

Aktivieren von Alarmaktionen

So aktivieren Sie Alarmaktionen für ein CloudWatch Alarm, rufen Sie die AmazonCloudWatchClient's an`enableAlarmActions`mit einemEnableAlarmActionsRequestenthält einen oder mehrere Namen von Alarmen, deren Aktionen Sie aktivieren möchten.

Importe

```
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;
```

Code

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

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

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

Deaktivieren von Alarmaktionen

So deaktivieren Sie Alarmaktionen für ein CloudWatch Alarm, rufen Sie die AmazonCloudWatchClient's and`disableAlarmActions`mit einem[DisableAlarmActionsRequest](#)enthält einen oder mehrere Namen von Alarmen, deren Aktionen Sie deaktivieren möchten.

Importe

```
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;
```

Code

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

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

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

Weitere Informationen

- Erstellen Sie Alarme, um eine Instance anzuhalten, zu beenden, neu zu starten oder wiederherzustellenimAmazon CloudWatch-Benutzerhandbuch
- PutMetricAlarmimAmazon CloudWatch-API-Referenz
- EnableAlarmActionsimAmazon CloudWatch-API-Referenz
- DisableAlarmActionsimAmazon CloudWatch-API-Referenz

Senden von Ereignissen an CloudWatch

CloudWatchEreignisse bieten einen Stream von Systemereignissen nahezu in Echtzeit, der Änderungen inAWSRessourcen zuAmazon EC2Instanzen,Lambda-Funktionen,Kinesis-Streams,Amazon ECS-Aufgaben,Step FunctionsZustandsautomaten,Amazon SNS-ThemenAmazon SQSWarteschlangen oder integrierte Ziele. Sie können Ereignisse zuordnen und sie zu einer oder mehreren Zielfunktionen oder Streams umleiten, indem Sie einfache Regeln nutzen.

Hinzufügen von Ereignissen

So fügen Sie Benutzerdefiniert hinzuCloudWatchevents, rufen Sie AmazonCloudWatchEventsClient'`sputEvents`-Methode mit einerPutEventsRequest-Objekt mit einer oder mehrerenPutEventsRequestEntry-Objekte, die Details zu jedem Ereignis enthalten. Sie können mehrere Parameter für den Eintrag angeben, wie z. B. die Quelle und den Typ des Ereignisses, mit dem Ereignis verknüpfte Ressourcen usw.

> **Note**
>
> Sie können maximal 10 Ereignisse pro Aufruf von `putEvents` angeben.

Importe

```
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;
```

Code

```
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);
```

Hinzufügen von Regeln

Sie erstellen oder Aktualisieren Sie einer Regel die AmazonCloudWatchEventsClients`putRule`-Methode mit einer`PutRuleRequest`mit dem Namen der Regel und optionalen Parametern wie`Ereignismuster`,IAMRolle, um mit der Regel zu verknüpfen, und ein`Ausdruck zeitlich planen`das beschreibt, wie oft die Regel ausgeführt wird.

Importe

```
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;
```

Code

```
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);
```

Hinzufügen von Zielen

Ziele sind die Ressourcen, die beim Auslösen einer Regel aufgerufen werden. Ziele können z. B. Amazon EC2-Instances, Lambda-Funktionen, Kinesis-Streams, Amazon ECS-Aufgaben, Step Functions-Zustandsautomaten sowie integrierte Ziele umfassen.

Sie fügen ein Ziel zu einer Regel hinzu, indem Sie die AmazonCloudWatchEventsClients aufrufen`putTargets`-Methode mit einerputTargetsRequestSie enthält die zu aktualisierende Regel und eine Liste der Ziele, die zu der Regel hinzugefügt werden sollen.

Importe

```
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;
```

Code

```
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);
```

Weitere Informationen

- Hinzufügen von Ereignissen mit PutEventsimAmazon CloudWatch Events-Benutzerhandbuch
- Planen von Ausdrücken für Regelnimamazon CloudWatch Events-Benutzerhandbuch
- Ereignistypen fürCloudWatchEventsimAmazon CloudWatch Events-Benutzerhandbuch

- Ereignisse und EreignismusterimAmazon CloudWatch Events-Benutzerhandbuch
- PutEventsimAmazon CloudWatch Events-API-Referenz
- PutTargetsimAmazon CloudWatch Events-API-Referenz
- PutRuleimAmazon CloudWatch Events-API-Referenz

DynamoDB-Beispiele unter Verwenden der AWS SDK for Java

Dieser Abschnitt bietet Beispiele für die Programmierung von DynamoDB mithilfe des AWS SDK for Java.

> **Note**
>
> Die Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der komplette Beispiel-Code steht auf GitHub bereit. Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

Themen

- Arbeiten mit Tabellen in DynamoDB
- Verwenden von Elementen in DynamoDB

Arbeiten mit Tabellen in DynamoDB

Tabellen sind Container für alle Elemente in einer DynamoDB-Datenbank. Bevor Sie Daten zu DynamoDB hinzufügen oder daraus entfernen können, müssen Sie eine Tabelle erstellen.

Für jede Tabelle definieren Sie:

- Einen Tabellennamen, der eindeutig für Ihr Konto und Ihre Region ist.
- Einen Primärschlüssel, für den jeder Wert eindeutig sein muss. Ihre Tabelle kann keine zwei Elemente mit demselben Primärschlüsselwert enthalten.

  Ein Primärschlüssel kann einfach sein, also aus einem Schlüssel mit einer einzigen Partition (HASH) bestehen, oder zusammengesetzt, also aus einer Partition und einem Sortierschlüssel (RANGE).

Jeder Schlüsselwert hat einen zugehörigen Datentyp, der von der ScalarAttributeTypeKlasse aufgezählt wird. Der Schlüsselwert kann binär (B), numerisch (n) oder eine Zeichenfolge (S) sein. Weitere Informationen finden Sie unter Benennungsregeln und Datentypen imAmazon DynamoDB Entwicklerhandbuch.

- Werte zum bereitgestellten Durchsatz, die die Anzahl der reservierten Lese-Schreib-Kapazitätseinheiten für die Tabelle angeben.

> **Note**
>
> Amazon DynamoDBDie Preisgestaltung basiert auf den bereitgestellten Durchsatzwerten, die Sie in Ihren Tabellen festgelegt haben. Reservieren Sie also nur so viel Kapazität, wie Sie für Ihre Tabelle benötigen.

Der bereitgestellte Durchsatz für eine Tabelle kann jederzeit geändert werden. So können Sie die Kapazität anpassen, wenn sich Ihre Anforderungen ändern.

Erstellen einer Tabelle

Verwenden Sie die`createTable` Methode des DynamoDBClients, um eine neueDynamoDB Tabelle zu erstellen. Sie müssen Tabellenattribute und ein Tabellenschema erstellen. Beide Komponenten fließen in den Primärschlüssel der Tabelle ein. Sie müssen auch anfänglich bereitgestellte Durchsatzwerte und einen Tabellennamen angeben. Definieren Sie nur wichtige Tabellenattribute beim Erstellen Ihrer DynamoDB-Tabelle.

> **Note**
>
> Wenn eine Tabelle mit dem von Ihnen ausgewählten Namen bereits existiert, AmazonServiceExceptionwird eine geworfen.

Importe

```
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;
```

Erstellen einer Tabelle mit einem einfachen Primärschlüssel

Dieser Code erstellt eine Tabelle mit einem einfachen Primärschlüssel ("Name").

Code

```
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);
}
```

Das vollständige Beispiel finden Sie aufGitHub.

Erstellen einer Tabelle mit einem zusammengesetzten Primärschlüssel

Füge ein weiteres AttributeDefinitionund KeySchemaElementzu hinzu CreateTableRequest.

Code

```
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);
```

Das vollständige Beispiel finden Sie aufGitHub.

Auflisten von Tabellen

Sie können die Tabellen in einer bestimmten Region auflisten, indem Sie die`listTables` Methode des DynamoDBClients aufrufen.

> **Note**
>
> Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert, ResourceNotFoundExceptionwird a ausgelöst.

Importe

```
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;
```

Code

```
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;
        }
```

Standardmäßig werden bis zu 100 Tabellen pro Aufruf zurückgegeben. Verwenden Sie diese Option`getLastEvaluatedTableName` für das zurückgegebene ListTablesResultObjekt, um die zuletzt ausgewertete Tabelle abzurufen. Mit diesem Wert können Sie die Auflistung nach dem zuletzt zurückgegebenen Wert der vorherigen Auflistung beginnen.

Das vollständige Beispiel finden Sie aufGitHub.

Beschreiben (Abrufen von Informationen zu) einer Tabelle

Rufen Sie die`describeTable` Methode des DynamoDBClients auf.

> **Note**
>
> Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert, ResourceNotFoundExceptionwird a ausgelöst.

Importe

```
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;
```

Code

```
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);
}
```

Das vollständige Beispiel finden Sie aufGitHub.

Ändern (Aktualisieren) einer Tabelle

Sie können die bereitgestellten Durchsatzwerte Ihrer Tabelle jederzeit ändern, indem Sie die`updateTable` Methode des DynamoDBClients aufrufen.

> **Note**
>
> Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert, ResourceNotFoundExceptionwird a ausgelöst.

Importe

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

Code

```
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);
}
```

Das vollständige Beispiel finden Sie aufGitHub.

Löschen einer Tabelle

Rufen Sie die`deleteTable` Methode des DynamoDBClients auf und übergeben Sie ihr den Namen der Tabelle.

> **Note**
>
> Wenn die benannte Tabelle für Ihr Konto und Ihre Region nicht existiert, ResourceNotFoundExceptionwird a ausgelöst.

Importe

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

Code

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

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

Das vollständige Beispiel finden Sie aufGitHub.

Weitere Infos

- Richtlinien für die Arbeit mit Tabellen imAmazon DynamoDB Entwicklerhandbuch
- Arbeiten mit TabellenDynamoDB imAmazon DynamoDB Entwicklerhandbuch

Verwenden von Elementen in DynamoDB

In DynamoDB ist ein Element eine Sammlung aus Attributen, von denen jedes über einen Namen und einen Wert verfügt. Ein Attributwert kann eine Skalarfunktion, eine Gruppe oder ein Dokumenttyp

sein. Weitere Informationen finden Sie unterBenennungsregeln und DatentypenimAmazon DynamoDBEntwicklerhandbuch.

Abrufen (empfangen) eines Elements aus einer Tabelle

Rufen Sie die AmazonDynamoDBs an`getItem`Methode und übergeben Sie es eingetItemRequest-Objekt mit dem Tabellennamen und dem Primärschlüsselwert des anzufordern Elements. Sie gibt ein GetItemResult-Objekt zurück.

Mit der `getItem()`-Methode des zurückgegebenen `GetItemResult`-Objekts können Sie eine Zuordnung aus Schlüssel-(Zeichenfolgen)- und Wert-(AttributeValue)-Paaren abrufen, die mit dem Element verknüpft sind.

Importe

```
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;
```

Code

```
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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Hinzufügen eines neuen Elements zu einer Tabelle

Erstellen Sie eine Map mit Schlüssel-Wert-Paaren, die die Attribute des Elements darstellen. Diese müssen Werte für die Primärschlüsselfelder der Tabelle enthalten. Wenn das Element mit dem Primärschlüssel bereits vorhanden ist, werden dessen Felder durch die Anforderung aktualisiert.

> **Note**
>
> Wenn die angegebene Tabelle für Ihr Konto und Ihre Region nicht vorhanden ist, wird eine ResourceNotFoundException ausgelöst.

Importe

```
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;
```

Code

```
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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Aktualisieren eines vorhandenen Elements in einer Tabelle

Sie können ein Attribut für ein Element, das bereits in einer Tabelle vorhanden ist, aktualisieren, indem Sie die AmazonDynamoDB`updateItem`-Methode, die einen Tabellennamen, Primärschlüsselwert und eine Map mit Feldern übergeben, die aktualisiert werden sollen.

> **Note**
>
> Wenn die angegebene Tabelle für Ihr Konto und Ihre Region oder das durch den übergebenen Primärschlüssel angegebene Element nicht vorhanden ist, wird eine ResourceNotFoundException ausgelöst.

Importe

```
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;
```

Code

```
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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Verwenden der DynamoDBMapper-Klasse

DieAWS SDK for Javastellt ein zur VerfügungDynamoDBMapper-Klasse, mit der Sie Ihre clientseitigen Klassen zuordnen könnenAmazon DynamoDBTische. So verwenden Sie denDynamoDBMapperdefinieren Sie die Beziehung zwischen Elementen in einemDynamoDB-Tabelle und ihre entsprechenden Objekt-Instances im Code mithilfe von Anmerkungen (wie im folgenden Codebeispiel gezeigt). Die DynamoDBMapper-Klasse ermöglicht Ihnen den Zugriff auf Ihre

Tabellen, das Ausführen verschiedener Create-, Read-, Update-, und Delete-Operationen (CRUD) und das Ausführen von Abfragen.

> **Note**
>
> Die DynamoDBMapper-Klasse erlaubt kein Erstellen, Aktualisieren oder Löschen von Tabellen.

Importe

```
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;
```

Code

Das folgende Java-Codebeispiel zeigt, wie Sie Inhalte zumMusikTabelle unter Verwendung desDynamoDBMapper-Klasse. Nachdem der Inhalt der Tabelle hinzugefügt wurde, beachten Sie, dass ein Element mithilfe der Schlüssel Partition und Sortieren geladen wird. Anschließend wird das Element Auszeichnungen aktualisiert. Weitere Informationen zum Erstellen desMusikTabelle, siheErstellen einer TabelleimAmazon DynamoDBEntwicklerhandbuch.

```
        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;
    }
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- Richtlinien für das Arbeiten mit ElementenimAmazon DynamoDBEntwicklerhandbuch
- Verwenden von Elementen inDynamoDBimAmazon DynamoDBEntwicklerhandbuch

Amazon EC2-Beispiele unter Verwenden der AWS SDK for Java

Dieser Abschnitt enthält Beispiele für die Programmierung vonAmazon EC2mit demAWS SDK for Javaaus.

Themen

- Tutorial: Starten einer EC2-Instance
- Verwenden von IAM-Rollen zum Gewähren von Zugriff aufAWSRessourcen aufAmazon EC2

- Tutorial: Amazon EC2 Spot-Instances
- Tutorial: AdvancedAmazon EC2Verwaltung von Spot-Anforderungen
- Verwalten von Amazon EC2-Instances
- Verwenden von Elastic IP-Adressen in Amazon EC2
- Verwenden von Regionen und Availability Zones
- Arbeiten mit Amazon EC2-Schlüsselpaaren
- Arbeiten mit Sicherheitsgruppen in Amazon EC2

Tutorial: Starten einer EC2-Instance

In diesem Tutorial wird gezeigt, wie Sie mit derAWS SDK for Javaum eine EC2-Instance zu starten.

Themen

- Voraussetzungen
- Erstellen einer Amazon EC2-Sicherheitsgruppe
- Erstellen eines Schlüsselpaares
- Führen Sie einAmazon EC2-Instance

Voraussetzungen

Bevor Sie beginnen, sollten Sie überprüfen, ob Sie ein erstellt habenAWS-Kontound dass Sie eingerichtet habenAWS-Anmeldeinformationen. Weitere Informationen finden Sie unter Erste Schritte mit .

Erstellen einer Amazon EC2-Sicherheitsgruppe

EC2-Classic geht in den Ruhestand

> **Warning**
>
> EC2-Classic wird am 15. August 2022 eingestellt. Wir empfehlen Ihnen die Migration von EC2-Classic zu einer VPC. Weitere Informationen finden Sie unterMigration von EC2-Classic zu einer VPCimAmazon EC2-Benutzerhandbuch für Linux-Instancesoder dasAmazon EC2 EC2-Benutzerhandbuch für Windows-Instances. Siehe auch den Blog-Posting.EC2-Classic-Classic-Networking geht in den Ruhestand — So bereiten Sie sich vor.

Erstellen Sie eine Sicherheitsgruppe, die als virtuelle Firewall agiert und den Netzwerkverkehr für eine oder mehrere EC2-Instances steuert. Standardmäßig weist Amazon EC2 Ihre Instances einer Sicherheitsgruppe zu, die keinen eingehenden Datenverkehr ermöglicht. Sie können eine Sicherheitsgruppe erstellen, die es den EC2-Instances ermöglicht, eingehenden Datenverkehr zu akzeptieren. Wenn Sie beispielsweise eine Verbindung zu einer Linux-Instance herstellen müssen, muss die Sicherheitsgruppe so konfiguriert werden, dass SSH-Datenverkehr möglich ist. Sie können eine Sicherheitsgruppe mithilfe der Amazon EC2-Konsole oder dem AWS SDK for Java erstellen.

Sie erstellen eine Sicherheitsgruppe entweder zur Verwendung in EC2-Classic oder EC2-VPC. Weitere Informationen zu EC2-Classic und EC2-VPC finden Sie unterUnterstützte PlattformenimAmazon EC2-Benutzerhandbuch für Linux-Instances.

Weitere Informationen zum Erstellen einer Sicherheitsgruppe mithilfe derAmazon EC2-Konsole finden Sie unterAmazon EC2SicherheitsgruppenimAmazon EC2-Benutzerhandbuch für Linux-Instances.

1. Erstellen und initialisieren Sie eineCreateSecurityGroupRequestsein. Verwenden derwithGroupNameMethode, um den Namen der Sicherheitsgruppe festzulegen, undwithDescription-Methode, um die Beschreibung der Sicherheitsgruppe wie folgt festzulegen:

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

   Der Name der Sicherheitsgruppe muss innerhalb derAWS-Region, in der Sie IhreAmazon EC2Client. Sie müssen US-ASCII-Zeichen für den Namen und die Beschreibung der Sicherheitsgruppe verwenden.

2. Übergeben Sie das Anforderungsobjekt als Parameter an dencreateSecurityGroup-Methode. Die Methode gibt eineCreateSecurityGroupResult-Objekt wie folgt:

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

   Wenn Sie versuchen, eine Sicherheitsgruppe mit dem gleichen Namen wie dem einer bereits vorhandenen Sicherheitsgruppe zu erstellen, gibt `createSecurityGroup` eine Ausnahme aus.

Standardmäßig erlaubt eine neue Sicherheitsgruppe keinen eingehenden Datenverkehr zur Amazon EC2-Instance. Um eingehenden Datenverkehr zu erlauben, müssen Sie ausdrücklich eingehende Daten für die Sicherheitsgruppe autorisieren. Sie können eingehende Daten für einzelne

IP-Adressen, für einen IP-Adressbereich, ein bestimmtes Protokoll sowie für TCP-/UDP-Ports autorisieren.

1. Erstellen und initialisieren Sie eineIpPermissionsein. Verwenden derMit IPv4-Bereichen-Methode, um den IP-Adressbereich festzulegen, von dem eingehende Daten autorisiert werden sollen. Durch Aufruf derwithIpProtocolMethode, um das IP-Protokoll festzulegen. Verwenden derwithFromPortundwithToPort-Methoden zum Angeben eines Portbereichs, von dem eingehende Daten autorisiert werden sollen, wie folgt:

```
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);
```

   Alle im `IpPermission`-Objekt angegebenen Bedingungen müssen erfüllt sein, damit eingehende Daten zugelassen werden.

   Geben Sie die IP-Adresse über CIDR-Notation an. Wenn Sie das Protokoll als TCP/UDP angeben, müssen Sie einen Quell- und Ziel-Port festlegen. Sie können Ports nur bei Angabe von TCP oder UDP autorisieren.

2. Erstellen und initialisieren Sie eineAuthorizeSecurityGroupIngressRequestsein. Verwenden der`withGroupName`-Methode, um den Namen der Sicherheitsgruppe anzugeben und`IpPermission`Objekt, das Sie zuvor initialisiert habenwithIpPermissions-Methode wie folgt:

```
AuthorizeSecurityGroupIngressRequest authorizeSecurityGroupIngressRequest =
    new AuthorizeSecurityGroupIngressRequest();

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

3. Übergeben Sie das Anforderung-Objekt anauthorizeSecurityGroupIngress-Methode wie folgt:

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

Wenn Sie `authorizeSecurityGroupIngress` mit IP-Adressen aufrufen, für die eingehende Daten bereits autorisiert sind, gibt diese Methode eine Ausnahme aus. Erstellen und initialisieren Sie ein neues `IpPermission`-Objekt, um eingehende Daten für andere IPs, Ports und Protokolle zu autorisieren, und rufen Sie dann `AuthorizeSecurityGroupIngress` auf.

Wann immer Sie denauthorizeSecurityGroupIngressoderauthorizeSecurityGroupEgress-Methoden wird eine Regel zur Sicherheitsgruppe hinzugefügt.

Erstellen eines Schlüsselpaares

Beim Start einer EC2-Instance müssen Sie ein Schlüsselpaar angeben. Anschließend geben Sie den privaten Schlüssel des Schlüsselpaars an, wenn Sie sich mit der Instance verbinden. Sie können ein Schlüsselpaar erstellen oder ein vorhandenes Schlüsselpaar verwenden, das Sie beim Start anderer Instances genutzt haben. Weitere Informationen finden Sie unterAmazon EC2SchlüsselpaareimAmazon EC2Benutzerhandbuch für Linux-Instances.

1. Erstellen und initialisieren Sie eine CreateKeyPairRequest-Instance. Verwenden Sie die withKeyName-Methode, um den Namen des Schlüsselpaars wie folgt festzulegen:

```
CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();

createKeyPairRequest.withKeyName(keyName);
```

> **Important**
>
> Namen von Schlüsselpaaren müssen eindeutig sein. Wenn Sie versuchen, ein Schlüsselpaar mit dem gleichen Namen wie dem eines bereits vorhandenen Schlüsselpaars zu erstellen, wird eine Ausnahme ausgelöst.

2. Übergeben Sie das Anforderungsobjekt an die createKeyPair-Methode. Die Methode gibt wie folgt eine CreateKeyPairResult-Instance zurück:

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

3. Rufen Sie die getKeyPair-Methode des resultierenden Objekts auf, um ein KeyPair-Objekt abzurufen. Rufen Sie das`KeyPair`-ObjektegetKeyMaterial-Methode, um den unverschlüsselten, PEM-kodierten privaten Schlüssel zu erhalten:

```
KeyPair keyPair = new KeyPair();

keyPair = createKeyPairResult.getKeyPair();

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

Führen Sie einAmazon EC2-Instance

Führen Sie die folgenden Schritte aus, um eine oder mehrere identisch konfigurierte EC2-Instances aus demselben Amazon Machine Image (AMI) zu starten. Nach dem Erstellen Ihrer EC2-Instances können Sie deren Status prüfen. Sobald die EC2-Instances ausgeführt werden, können Sie sich mit ihnen verbinden.

1. Erstellen und initialisieren Sie eine RunInstancesRequest-Instance. Stellen Sie sicher, dass das AMI, das Schlüsselpaar und die Sicherheitsgruppe, die Sie angeben, in der beim Erstellen des Client-Objekts angegebenen Region vorhanden sind.

```
RunInstancesRequest runInstancesRequest =
   new RunInstancesRequest();

runInstancesRequest.withImageId("ami-a9d09ed1")
                   .withInstanceType(InstanceType.T1Micro)
                   .withMinCount(1)
                   .withMaxCount(1)
                   .withKeyName("my-key-pair")
                   .withSecurityGroups("my-security-group");
```

withImageId

- Die ID des AMI. Weitere Informationen, wie Sie öffentliche AMIs von Amazon finden oder selbst erstellen, finden Sie unter Amazon Machine Image (AMI).

withInstanceType

- Ein Instance-Typ, der kompatibel mit dem angegebenen AMI ist. Weitere Informationen finden Sie unterInstance-TypenimAmazon EC2Benutzerhandbuch für Linux-Instances

withMinCount

- Die Mindestanzahl zu startender EC2-Instances. Wenn dies mehr Instances sind, als Amazon EC2 in der gewünschten Availability Zone starten kann, startet Amazon EC2 keine Instances.

withMaxCount

- Die Höchstanzahl zu startender EC2-Instances. Wenn dies mehr Instances sind, als Amazon EC2 in der gewünschten Availability Zone starten kann, startet Amazon EC2 die höchstmögliche Anzahl an Instances über `MinCount`. Sie können zwischen 1 und der maximalen Anzahl der Instances starten, die für Sie beim jeweiligen Instance-Typ zulässig sind. Weitere Informationen finden Sie unter Wie viele Instances kann ich ausführenAmazon EC2imAmazon EC2Allgemeine Häufig gestellte Fragen

withKeyName

- Der Name des EC2-Schlüsselpaars. Wenn Sie eine Instance ohne Angabe eines Schlüsselpaars starten, können Sie sich nicht mit ihr verbinden. Weitere Informationen finden Sie unter Erstellen eines Schlüsselpaars.

withSecurityGroups

- Eine oder mehrere Sicherheitsgruppen. Weitere Informationen finden Sie unterErstellen einesAmazon EC2Sicherheitsgruppeaus.

2. Starten Sie die Instances, indem Sie das Anforderungsobjekt an die runInstances-Methode übergeben. Die Methode gibt ein RunInstancesResult-Objekt wie folgt zurück:

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

Sobald die Instance ausgeführt wird, können Sie sich mit dem Schlüsselpaar mit ihr verbinden. Weitere Informationen finden Sie unterHerstellen einer Verbindung mit Ihrer Linux-Instance. imAmazon EC2Benutzerhandbuch für Linux-Instances

Verwenden von IAM-Rollen zum Gewähren von Zugriff aufAWSRessourcen aufAmazon EC2

Alle Anforderungen anAmazon Web Services(AWS) muss kryptografisch mit den Anmeldeinformationen signiert sein, die vonAWSaus. Sie könnenIAM rolesum bequem sicheren Zugriff zu gewährenAWSRessourcen von IhremAmazon EC2Instanzen.

In diesem Thema finden Sie Informationen zur Verwendung von IAM-Rollen mit Java SDK-Anwendungen aufAmazon EC2aus. Weitere Informationen zu IAM-Instances finden Sie unterIAM-Rollen fürAmazon EC2imAmazon EC2-Benutzerhandbuch für Linux-Instances

Die standardmäßige Anbieterkette und EC2-Instance-Profile

Wenn Ihre Anwendung eine erstelltAWS-Client mit dem Standardkonstruktor sucht der Client mit demAnbieterkette für Standarddatenin der folgenden Reihenfolge:

1. in den Java-Systemeigenschaften: `aws.accessKeyId` und `aws.secretKey`
2. in System-Umgebungsvariablen: `AWS_ACCESS_KEY_ID` und `AWS_SECRET_ACCESS_KEY`
3. in der Standarddatei für Anmeldeinformationen (der Speicherort dieser Datei hängt von der jeweiligen Plattform ab)
4. Anmeldeinformationen, die über dieAmazon EC2Container Service, wenn der`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`Die Umgebungsvariable ist festgelegt und der Sicherheitsmanager verfügt über die Berechtigung zum Zugriff auf die Variable.
5. in den Anmeldeinformationen des Instance-Profils, die zu den Instance-Metadaten gehören, die mit der IAM-Rolle für die EC2-Instance verknüpft sind
6. Web-Identitätstoken-Anmeldeinformationen aus der Umgebung oder dem Container.

Der Schritt Anmeldeinformationen des Instance-Profils in der standardmäßigen Anbieterkette ist nur verfügbar, wenn Ihre Anwendung auf einer Amazon EC2-Instance ausgeführt wird. Er ist allerdings am einfachsten und bietet die beste Sicherheit beim Umgang mit Amazon EC2-Instances. Sie können auch eine InstanceProfileCredentialsProvider-Instance direkt an den Client-Konstruktor übergeben und erhalten so Anmeldeinformationen des Instance-Profils ohne Verarbeitung der gesamten standardmäßigen Anbieterkette.

Beispiel:

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

Wenn Sie diesen Ansatz verwenden, ruft das SDK temporär abAWSAnmeldeinformationen mit den gleichen Berechtigungen wie diejenigen der IAM-Rolle, die mit derAmazon EC2-Instance in seinem Instance-Profil. Obwohl diese Anmeldeinformationen vorübergehend sind und schließlich ablaufen würden,`InstanceProfileCredentialsProvider`Aktualisiert sie in regelmäßigen Abständen für Sie, damit die erhaltenen Anmeldeinformationen weiterhin aufAWSaus.

> **Important**
>
> Die Anmeldeinformationen werden nur dann automatisch aktualisiert, wenn Sie den standardmäßigen Client-Konstruktor verwenden, der seinen eigenen `InstanceProfileCredentialsProvider` als Teil der standardmäßigen Anbieterkette erstellt, oder wenn Sie eine `InstanceProfileCredentialsProvider`-Instance direkt an den Client-Konstruktor übergeben. Wenn Sie Anmeldeinformationen des Instance-Profils auf andere Weise abrufen oder übergeben, sind Sie selbst für die Überprüfung und ggf. für die Aktualisierung abgelaufener Anmeldeinformationen zuständig.

Wenn der Client-Konstruktor keine Anmeldeinformationen mithilfe der standardmäßigen Anbieterkette von Anmeldeinformationen finden kann, wird eine AmazonClientException ausgelöst.

Vorgehensweise: Verwenden von IAM-Rollen für EC2-Instances

Die folgende Anleitung zeigt, wie Sie ein Objekt von abrufen könnenAmazon S3Verwenden einer IAM-Rolle zum Verwalten des Zugriffs.

Erstellen einer IAM-Rolle

Erstellen Sie eine IAM-Rolle, die Lesezugriff auf Amazon S3 gewährt.

1. Öffnen Sie die IAM-Konsole.
2. Wechseln Sie im Navigationsbereich zu Roles (Rollen) und klicken Sie auf Create New Role (Neue Rolle erstellen).
3. Geben Sie einen Namen für die Rolle ein und klicken Sie dann auf Next Step. Notieren Sie sich diesen Namen, da Sie ihn benötigen, wenn Sie die Amazon EC2-Instance starten.
4. Auf derRollentyp wählenseite, unter AWS-ServiceRollenSelect Amazon EC2 aus.
5. Auf derBerechtigungen festlegenseite, unterWählen Sie eine RichtlinienvorlageSelect Amazon S3Schreibgeschützter Zugriff, dannNächster Schrittaus.
6. Wählen Sie auf der Seite Review (Prüfen) Create Role (Rolle erstellen) aus.

Starten einer EC2-Instance und Angeben der IAM-Rolle

Sie können einAmazon EC2-Instance mit einer IAM-Rolle mit derAmazon EC2-Konsole oder dasAWS SDK for Javaaus.

- So starten Sie einAmazon EC2Befolgen Sie mit der -Konsole die Anweisungen unterErste Schritte mitAmazon EC2Linux InstancesimAmazon EC2-Benutzerhandbuch für Linux-Instances

  Wenn Sie die Seite Review Instance Launch erreichen, klicken Sie auf Edit instance details. In :IAM-RolleWählen Sie die zuvor erstellte IAM-Rolle aus. Befolgen Sie die Anweisungen und schließen Sie den Vorgang ab.

  > **Note**
  >
  > Zum Herstellen einer Verbindung mit der Instance müssen Sie eine Sicherheitsgruppe und ein Schlüsselpaar neu erstellen oder vorhandene Anmeldeinformationen auswählen.

- So starten Sie einAmazon EC2-Instance mit einer IAM-Rolle mit derAWS SDK for Javafinden Sie unterAusführen eines sAmazon EC2-Instanceaus.

Erstellen Ihrer Anwendung

Lassen Sie uns die Beispielanwendung zum Ausführen auf der EC2-Instance erstellen. Zuerst erstellen Sie ein Verzeichnis, in dem Sie die Tutorial-Dateien abspeichern können (z. B. `GetS3ObjectApp`).

Kopieren Sie als Nächstes die AWS SDK for Java-Bibliotheken in Ihr neu erstelltes Verzeichnis. Wenn Sie das AWS SDK for Java in Ihr `~/Downloads`-Verzeichnis heruntergeladen haben, können Sie sie mit folgenden Befehlen kopieren:

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

Legen Sie eine neue Datei namens `GetS3Object.java` an und fügen Sie den folgenden Code ein:

```
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();
  }
}
```

Legen Sie eine weitere neue Datei namens `build.xml` an und fügen Sie folgende Zeilen ein:

```
<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>
```

Erstellen und starten Sie das geänderte Programm. Beachten Sie, dass keine Anmeldeinformationen im Programm gespeichert werden. Deshalb, es sei denn, Sie haben IhreAWSAnmeldeinformationen, die bereits angegeben sind, löst der Code`AmazonServiceException`aus. Beispiel:

```
$ 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
```

Übertragen des kompilierten Programms an Ihre EC2-Instance

Übertragen Sie das Programm per SCP (Secure Copy, Amazon EC2) zusammen mit den -Bibliotheken auf Ihre AWS SDK for Java-Instance. Die Reihenfolge der Befehle sieht in etwa wie folgt aus:

```
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**
>
> Abhängig von der verwendeten Linux-Distribution lautet der user name (Benutzername) "ec2-user", "root" oder "ubuntu". Sie können den öffentlichen DNS-Namen Ihrer Instance abrufen, indem Sie die EC2-Konsole öffnen und den Wert Public DNS auf der Registerkarte Description beachten (z. B. `ec2-198-51-100-1.compute-1.amazonaws.com`).

Bei den vorhergehenden Befehlen:

- ist `GetS3Object.class` Ihr kompiliertes Programm,
- `build.xml` ist die Ant-Datei zum Erstellen und Ausführen Ihres Programms und
- die Verzeichnisse `lib` und `third-party` sind die entsprechenden Bibliotheksordner aus dem AWS SDK for Java.
- Der Schalter `-r` gibt an, dass `scp` sämtliche Inhalte der Verzeichnisse `library` und `third-party` in der AWS SDK for Java-Distribution rekursiv kopieren sollte.
- Der Schalter `-p` sorgt dafür, dass `scp` die Berechtigungen der Quelldateien beibehalten soll, während diese zum Ziel kopiert werden.

> **Note**
>
> Die `-p`switch funktioniert nur unter Linux, macOS oder Unix. Wenn Sie Dateien von Windows kopieren, müssen Sie die Dateiberechtigungen auf Ihrer Instance mit dem folgenden Befehl korrigieren:

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

Ausführen des Beispielprogramms auf der EC2-Instance

Verbinden Sie sich zum Ausführen des Programms mit der Amazon EC2-Instance. Weitere Informationen finden Sie unterHerstellen einer Verbindung mit Ihrer Linux-InstanceimAmazon EC2-Benutzerhandbuch für Linux-Instances

Wenn **`ant`** auf Ihrer Instance nicht verfügbar ist, installieren Sie es mit folgendem Befehl:

```
sudo yum install ant
```

Führen Sie das Programm dann mithilfe von `ant` wie folgt aus:

```
ant run
```

Das Programm schreibt den Inhalt Ihres Amazon S3-Objekts in Ihr Befehlsfenster.

Tutorial: Amazon EC2 Spot-Instances

Übersicht

Mit Spot-Instances können Sie auf ungenutzte Amazon Elastic Compute Cloud (Amazon EC2)-Kapazität auf bis zu 90 % im Vergleich zum On-Demand-Instance-Preis bieten und alle erworbenen Instances so lange ausführen, wie Ihr Gebot den aktuellen Spot-Preis übersteigt. Amazon EC2 ändert den Spot-Preis regelmäßig abhängig von Angebot und Nachfrage. Kunden, deren Gebote den Spot-Preis erreichen oder übersteigen, erhalten Zugriff auf die verfügbaren Spot-Instances. Wie mit On-Demand-Instances und Reserved Instances erhalten Sie mit Spot-Instances eine weitere Möglichkeit, mehr Rechenkapazität zu erhalten.

Spot-Instances können die Amazon EC2-Kosten für Stapelverarbeitung, wissenschaftliche Forschung, Bildverarbeitung, Videocodierung, Daten- und Web-Crawling, Finanzanalysen und Tests spürbar senken. Darüber hinaus ermöglichen Spot-Instances Zugriff auf große Mengen an zusätzlicher Kapazität in Situationen, in denen diese Kapazität nicht dringend benötigt wird.

Senden Sie zur Nutzung von Spot-Instances eine Spot-Instance-Anforderung und geben Sie den Höchstpreis an, den Sie pro Instance-Stunde zu zahlen bereit sind; dies ist Ihr Gebot. Wenn Ihr Höchstgebot den aktuellen Spot-Preis übersteigt, wird Ihrer Anforderung stattgegeben. Ihre Instances werden so lange ausgeführt, bis Sie sie entweder beenden oder bis der Spot-Preis Ihr Höchstgebot übersteigt.

Wichtiger Hinweis:

- Sie zahlen oft weniger pro Stunde als Ihr Gebot. Amazon EC2 passt den Spot-Preis regelmäßig an, während Anfragen eingehen und die Verfügbarkeit schwankt. Alle Benutzer bezahlen für diesen Zeitraum denselben Spot-Preis, unabhängig davon, ob ihr Gebot höher lag. Daher zahlen Sie möglicherweise weniger als Ihr Gebot, aber Sie zahlen nie mehr als das Gebot.
- Wenn Sie Spot-Instances ausführen und Ihr Gebot nicht mehr mindestens dem aktuellen Spot-Preis entspricht, werden Ihre Instances beendet. Daher sollten Sie sicherstellen, dass Ihre Workloads und Anwendungen flexibel genug sind, um aus dieser gelegentlichen Kapazität Nutzen zu ziehen.

Bei der Ausführung liefern Spot-Instances genau dieselbe Leistung wie andere Amazon EC2-Instances. Sie lassen sich auch genau so beenden wie andere Amazon EC2-Instances, wenn Sie sie nicht mehr benötigen. Beim Beenden einer Instance zahlen Sie für die angefangene Stunde (wie bei On-Demand- und Reserved Instances). Steigt der Spot-Preis allerdings über Ihr Gebot und Ihre Instance wird von Amazon EC2 beendet, wird Ihnen keine angefangene Nutzungsstunde in Rechnung gestellt.

In dieser Anleitung erfahren Sie, wie Sie mit AWS SDK for Java die folgenden Aufgaben ausführen:

- Senden einer Spot-Anfrage
- Ermitteln, wann die Spot-Anfrage erfüllt wird
- Abbrechen der Spot-Anfrage
- Beenden von dazugehörigen Instances

Voraussetzungen

Um diese Anleitung nutzen zu können, sollten Sie das AWS SDK for Java installiert haben und die grundlegenden Installationsvoraussetzungen erfüllen. Siehe .Einrichten derAWS SDK for JavaWeitere Informationen finden Sie unter.

Schritt 1: Einrichten Ihrer Anmeldeinformationen

Um dieses Codebeispiel verwenden zu können, müssen Sie eine EinrichtungAWS-Anmeldeinformationen. Siehe .EinrichtenAWSAnmeldeinformationen und Region für die EntwicklungEine entsprechende Anleitung finden Sie unter.

> **Note**
>
> Wir empfehlen, dass Sie die Anmeldeinformationen eines IAM-Benutzers für diese Werte nutzen. Weitere Informationen finden Sie unterRegistrieren beiAWSErstellen eines IAM-Benutzersaus.

Nachdem Sie Ihre Einstellungen eingerichtet haben, können Sie mit dem Beispiel-Code loslegen.

Schritt 2: Einrichten einer Sicherheitsgruppe

Eine Sicherheitsgruppe agiert als Firewall, die den zulässigen Verkehr zu und von einer Gruppe Instances steuert. Standardmäßig wird eine Instance ohne eine Sicherheitsgruppe gestartet.

Sämtlicher eingehender IP-Datenverkehr auf allen TCP-Ports wird daher verweigert. Vor dem Absenden unserer Spot-Anforderung richten wir also eine Sicherheitsgruppe ein, die den nötigen Netzwerkverkehr zulässt. Für die Zwecke dieser Anleitung erstellen wir eine neue Sicherheitsgruppe mit dem Namen "GettingStarted", mit der Secure Shell (SSH)-Datenverkehr von der IP-Adresse, von der Sie die Anwendung ausführen, zugelassen wird. Zur Einrichtung einer neuen Sicherheitsgruppe sollten Sie das folgende Codebeispiel einschließen oder ausführen. Dadurch wird die Sicherheitsgruppe per Programm eingerichtet.

Nach dem Erstellen eines `AmazonEC2`-Client-Objekts legen wir ein `CreateSecurityGroupRequest`-Objekt mit dem Namen "GettingStarted" und eine Beschreibung für die Sicherheitsgruppe an. Anschließend wird die `ec2.createSecurityGroup`-API zum Erstellen der Gruppe aufgerufen.

Zum Aktivieren des Zugriffs auf die Gruppe erstellen wir ein `ipPermission`-Objekt, bei dem der IP-Adressbereich des Subnetzes auf die CIDR-Darstellung des Subnetzes des lokalen Computers festgelegt ist. Das Suffix "/10" bei der IP-Adresse zeigt das Subnetz für die angegebene IP-Adresse an. Anschließend konfigurieren wir auch das `ipPermission`-Objekt mit dem TCP-Protokoll und dem Port 22 (SSH). Im letzten Schritt wird `ec2.authorizeSecurityGroupIngress` mit dem Namen der Sicherheitsgruppe und dem `ipPermission`-Objekt aufgerufen.

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

Hinweis: Sie müssen diese Anwendung nur einmal ausführen, um eine neue Sicherheitsgruppe zu erstellen.

Sie können die Sicherheitsgruppe auch mithilfe von AWS Toolkit for Eclipse erstellen.
Siehe .Verwalten von SicherheitsgruppenAWS Cost ExplorerWeitere Informationen finden Sie unter.

Schritt 3: Senden Ihrer Spot-Anforderung

Zum Senden einer Spot-Anforderung sollten Sie zuerst den Instance-Typ, das Amazon Machine Image (AMI) und den Höchstpreis für Ihr Gebot festlegen. Sie müssen auch die zuvor konfigurierte Sicherheitsgruppe aufnehmen, damit Sie sich bei Bedarf bei der Instance anmelden können.

Sie können zwischen verschiedenen Instance-Typen wählen.Amazon EC2Instance-Typen für eine vollständige Liste. Für diese Anleitung verwenden wir t1.micro, den günstigsten verfügbaren Instance-

Typ. Als Nächstes bestimmen wir, welches AMI wir nutzen möchten. Hier verwenden wir ami-a9d09ed1, das aktuelle Amazon Linux-AMI, das zum Erstellungszeitpunkt dieser Anleitung verfügbar war. Von Zeit zu Zeit kann es neuere AMIs geben. Die jeweils neueste AMI-Version lässt sich mit folgenden Schritten ermitteln:

1. Öffnen Sie die Amazon EC2-Konsole.
2. Wählen Sie die Schaltfläche Launch Instance (Instance starten).
3. Das erste Fenster zeigt die verfügbaren AMIs an. Die AMI-ID ist neben dem jeweiligen AMI-Titel aufgelistet. Alternativ können Sie die `DescribeImages`-API nutzen. Die Nutzung dieses Befehls wird in dieser Anleitung allerdings nicht behandelt.

Es gibt viele Wege zur Gebotsgestaltung für Spot-Instances. Eine gute Übersicht der unterschiedlichen Ansätze finden Sie im Video Bieten für Spot-Instances. Allerdings beschreiben wir drei allgemeine Strategien für den Einstieg: Gebote, um sicherzustellen, dass die Kosten geringer sind als bei On-Demand-Preisen; Gebote basierend auf dem Wert der resultierenden Berechnung; Gebote, um Rechenkapazität so schnell wie möglich zu erwerben.

- Senkung der Kosten unter On-Demand Sie haben eine Stapelverarbeitungsaufgabe, die einige Stunden oder Tage laufen wird. Allerdings sind Sie flexibel, was den Start und Abschluss angeht. Sie möchten die Aufgabe nach Möglichkeit günstiger als mit On-Demand-Instances abschließen. Sie nehmen den Verlauf der Spot-Preise für Instance-Typen entweder mit der AWS Management Console oder mit der Amazon EC2-API unter die Lupe. Weitere Informationen finden Sie unter Anzeigen des Spot-Preisverlaufs. Nachdem Sie den Preisverlauf für Ihren gewünschten Instance-Typ in einer bestimmten Availability Zone analysiert haben, gibt es zwei alternative Ansätze für Ihr Gebot:
  - Sie könnten ein Gebot am oberen Ende der Skala für Spot-Preise abgeben (aber immer noch unter dem On-Demand-Preis) und voraussehen, dass Ihre einmalige Spot-Anforderung sehr wahrscheinlich erfüllt wird und die Instance lange genug aktiv bleibt, um die Aufgabe abzuschließen.
  - Oder Sie können den Betrag, den Sie für Spot-Instances zu zahlen bereit sind, als % des On-Demand-Instance-Preises angeben und planen, viele Instances zu kombinieren, die im Laufe der Zeit über eine persistente Anforderungen gestartet wurden. Wenn der angegebene Preis überschritten wird, wird die Spot-Instance beendet. (Später in dieser Anleitung zeigen wir Ihnen, wie Sie diese Aufgabe automatisieren können.)
- Nicht mehr zahlen, als das Ergebnis einbringt Sie haben eine Aufgabe zur Datenverarbeitung, die ausgeführt werden soll. Sie kennen den Wert der Ergebnisse des Auftrags gut genug, um zu

wissen, wie sich der Wert als Rechenkosten darstellen lässt. Sie analysieren den Spot-Preisverlauf Ihres Instance-Typs und entscheiden sich dann für einen Gebotspreis, der sicherstellt, dass die Kosten der Rechenzeit geringer sind als der Wert der Auftragsergebnisse. Sie erstellen ein persistentes Gebot und lassen es zwischenzeitlich laufen, sobald der Spot-Preis Ihr Gebot erreicht oder darunter sinkt.

- Schneller Erwerb von Rechenkapazität Sie haben einen plötzlichen, kurzfristigen Bedarf an zusätzlicher Kapazität, die von On-Demand-Instances nicht bereitgestellt werden kann. Sie analysieren den Spot-Preisverlauf Ihres Instance-Typs und bieten dann über dem höchsten bisherigen Preis. So sorgen Sie dafür, dass Ihre Anforderung mit hoher Wahrscheinlichkeit schnell erfüllt wird und bis zum Abschluss der Aufgabe läuft.

Nachdem Sie den Gebotspreis ausgewählt haben, können Sie eine Spot-Instance anfordern. Für diese Anleitung bieten wir den On-Demand-Preis (0,03 USD) und maximieren so die Chancen, dass das Gebot erfüllt wird. Sie können die Typen verfügbarer Instances sowie die On-Demand-Preise für Instances ermitteln, indem Sie einenAmazon EC2Seite „Preisgestaltung“. Für die Dauer der Ausführung der Instances zahlen Sie bei Spot-Instances den bei der Anforderung angegebenen Stundensatz. Die Spot-Instance-Preise werden von Amazon EC2 festgelegt und ändern sich schrittweise entsprechend der langfristigen Trends bei Angebot und Nachfrage zu Spot-Instance-Kapazitäten. Sie können auch den Betrag, den Sie für eine Spot-Instance zu zahlen bereit sind, als % des On-Demand-Instance-Preises angeben. Um eine Spot-Instance anzufordern, müssen Sie Ihre Anfrage einfach mit den zuvor ausgewählten Parametern erstellen. Als Erstes erstellen wir ein `RequestSpotInstanceRequest`-Objekt. Für das Anforderungsobjekt sind die Anzahl der zu startenden Instances sowie der Gebotspreis nötig. Außerdem müssen Sie die `LaunchSpecification` für die Anforderung festlegen. Sie umfasst den Instance-Typ, die AMI-ID sowie die Sicherheitsgruppe, die Sie verwenden möchten. Sobald die Anforderung vorbereitet ist, rufen Sie die Methode `requestSpotInstances` des Objekts `AmazonEC2Client` auf. Das folgende Beispiel zeigt, wie Sie eine Spot-Instance anfordern.

```
// 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);
```

Beim Ausführen dieses Codes wird eine neue Spot-Instance-Anforderung ausgeführt. Mit weiteren Optionen können Sie Spot-Anforderungen konfigurieren. Um mehr zu erfahren, besuchen Sie bitteTutorial: AdvancedAmazon EC2Verwaltung der Spot-Anforderungenoder dasRequestSpotInstances-Klasse imAWS SDK for Java-API-Referenz.

> **Note**
>
> Sie zahlen für Spot-Instances, die tatsächlich gestartet werden. Achten Sie also darauf, getätigte Anforderungen zu stornieren und gestartete Instances zu beenden, damit Ihnen keine unnötigen Kosten entstehen.

Schritt 4: Ermitteln des Status Ihrer Spot-Anforderung

Als Nächstes erstellen wir einen Code, der darauf wartet, dass die Spot-Anforderung den Status "active" erreicht, bevor wir zum letzten Schritt wechseln. Zur Ermittlung des Status unserer Spot-Anforderung rufen wir die Methode describeSpotInstanceRequests ab und lesen den Status der Spot-Anforderungs-ID aus, die wir überwachen möchten.

Die in Schritt 2 erstellte Anforderungs-ID ist in der Antwort auf unsere `requestSpotInstances`-Anforderung enthalten. Im folgenden Beispielcode wird gezeigt, wie sich Anforderungs-IDs aus der `requestSpotInstances`-Antwort auslesen und zur Vorbereitung einer `ArrayList` nutzen lassen.

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

Rufen Sie zur Überwachung Ihrer Anforderungs-ID die Methode `describeSpotInstanceRequests` auf und ermitteln Sie so den Status der Anforderung. Warten Sie dann in einer Schleife, bis die Anforderung nicht mehr den Zustand "open" aufweist. Hinweis: Wir überprüfen hier, ob der Zustand ungleich "open" ist, anstatt etwa auf "active" zu überprüfen. Der Grund ist, dass die Anforderung direkt zu "closed" übergehen kann, wenn ein Problem mit den Argumenten der Anforderung vorliegt. Im folgenden Codebeispiel sehen Sie im Detail, wie diese Aufgabe umgesetzt wird.

```
// 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);
```

Nachdem dieser Code ausgeführt wird, ist Ihre Spot-Instance-Anforderung entweder abgeschlossen oder mit einem Fehler gescheitert, der auf dem Bildschirm angezeigt wird. In beiden Fällen können wir mit dem nächsten Schritt fortfahren, alle aktiven Anforderungen bereinigen und alle laufenden Instances beenden.

Schritt 5: Bereinigen der Spot-Anforderungen und -Instances

Schließlich müssen wir unsere Anforderungen und Instances bereinigen. Es ist wichtig, sowohl ausstehende Anforderungen zu stornieren als auch etwaige Instances zu beenden. Wenn Sie nur die Anforderungen stornieren, werden Ihre Instances nicht beendet und Sie müssen weiter für sie zahlen.

Wenn Sie die Instances beenden, können Ihre Spot-Anforderungen storniert werden. In einigen Fällen – etwa dann, wenn Sie persistente Gebote nutzen –, reicht das Beenden Ihrer Instances nicht aus, damit Ihre Anforderung nicht erneut erfüllt wird. Daher ist es eine bewährte Methode, sowohl die aktiven Gebote zu stornieren als auch alle laufenden Instances zu beenden.

Im folgenden Beispiel wird gezeigt, wie Sie Ihre Anforderungen stornieren.

```
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());
}
```

Zum Stornieren ausstehender Anforderungen brauchen Sie die jeweilige Instance-ID. Sie ist mit der Anforderung verknüpft, die sie gestartet hat. Im folgenden Codebeispiel ergänzen wir unseren Originalcode zur Überwachung der Instances um eine `ArrayList`. Darin speichern wir die Instance-ID aus der `describeInstance`-Antwort.

```
// 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);
```

Mithilfe der in der `ArrayList` gespeicherten Instance-IDs können Sie laufende Instances mit dem folgenden Codeausschnitt beenden:

```
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());
}
```

Zusammenfassung

Abschließend erstellen wir eine verbesserte objektorientierte Version. Sie kombiniert die zuvor gezeigten Schritte: Initialisierung des EC2-Clients; Senden der Spot-Anforderung; Ermittlung, wann die Spot-Anforderungen nicht mehr den Zustand "offen" aufweisen; sowie schließlich die Bereinigung ausstehender Spot-Anforderungen und zugehöriger Instances. Wir erstellen eine Klasse namens `Requests`, die diese Aktionen ausführt.

Zudem legen wir eine Klasse `GettingStarted`App mit einer main-Methode an, in der wir die High-Level-Funktionsaufrufe durchführen. Insbesondere initialisieren wir das zuvor beschriebene `Requests`-Objekt. Wir senden die Spot-Instance-Anforderung. Anschließend warten wir, bis die Spot-Anforderung den Zustand "active" erreicht. Schließlich bereinigen wir die Anforderungen und Instances.

Den vollständigen Quelltext dieses Beispiels finden Sie zum Lesen und Herunterladen auf GitHub.

Herzlichen Glückwunsch! Sie haben jetzt die Erste-Schritte-Anleitung zur Entwicklung von Spot-Instance-Software mit dem AWS SDK for Java abgeschlossen.

Nächste Schritte

Fahren Sie mit fortTutorial: AdvancedAmazon EC2Verwaltung der Spot-Anforderungenaus.

Tutorial: AdvancedAmazon EC2Verwaltung von Spot-Anforderungen

Mit Amazon EC2-Spot-Instances können Sie auf ungenutzte Amazon EC2-Kapazität bieten und diese Instances so lange ausführen, wie Ihr Gebot den aktuellen Spot-Preis übersteigt. Amazon EC2 ändert den Spot-Preis regelmäßig gemäß Angebot und Nachfrage. Weitere Informationen zu Spot-Instances finden Sie unter.Spot-InstancesimAmazon EC2-Benutzerhandbuch für Linux-Instances.

Voraussetzungen

Um diese Anleitung nutzen zu können, sollten Sie das AWS SDK for Java installiert haben und die grundlegenden Installationsvoraussetzungen erfüllen. Siehe .Einrichten derAWS SDK for JavaWeitere Informationen finden Sie unter.

Einrichten Ihrer Anmeldeinformationen

Zur Verwendung dieses Codebeispiels müssen SieAWS-Anmeldeinformationen.
Siehe .EinrichtenAWSAnmeldeinformationen und Region für die EntwicklungEine entsprechende Anleitung finden Sie unter.

> **Note**
>
> Wir empfehlen, dass Sie die Anmeldeinformationen einesIAMBenutzer, um diese Werte bereitzustellen. Weitere Informationen finden Sie unterRegistrieren beiAWSund erstelle einIAMBenutzeraus.

Nachdem Sie Ihre Einstellungen eingerichtet haben, können Sie mit dem Beispiel-Code loslegen.

Einrichten einer Sicherheitsgruppe

Eine Sicherheitsgruppe agiert als Firewall, die den zulässigen Verkehr zu und von einer Gruppe Instances steuert. Standardmäßig wird eine Instance ohne eine Sicherheitsgruppe gestartet. Sämtlicher eingehender IP-Datenverkehr auf allen TCP-Ports wird daher verweigert. Vor dem Absenden unserer Spot-Anforderung richten wir also eine Sicherheitsgruppe ein, die den nötigen Netzwerkverkehr zulässt. Für die Zwecke dieser Anleitung erstellen wir eine neue Sicherheitsgruppe mit dem Namen "GettingStarted", mit der Secure Shell (SSH)-Datenverkehr von der IP-Adresse, von der Sie die Anwendung ausführen, zugelassen wird. Zur Einrichtung einer neuen Sicherheitsgruppe sollten Sie das folgende Codebeispiel einschließen oder ausführen. Dadurch wird die Sicherheitsgruppe per Programm eingerichtet.

Nach dem Erstellen eines `AmazonEC2`-Client-Objekts legen wir ein `CreateSecurityGroupRequest`-Objekt mit dem Namen "GettingStarted" und eine Beschreibung für die Sicherheitsgruppe an. Anschließend wird die `ec2.createSecurityGroup`-API zum Erstellen der Gruppe aufgerufen.

Zum Aktivieren des Zugriffs auf die Gruppe erstellen wir ein `ipPermission`-Objekt, bei dem der IP-Adressbereich des Subnetzes auf die CIDR-Darstellung des Subnetzes des lokalen Computers

festgelegt ist. Das Suffix "/10" bei der IP-Adresse zeigt das Subnetz für die angegebene IP-Adresse an. Anschließend konfigurieren wir auch das `ipPermission`-Objekt mit dem TCP-Protokoll und dem Port 22 (SSH). Im letzten Schritt wird `ec2 .authorizeSecurityGroupIngress` mit dem Namen der Sicherheitsgruppe und dem `ipPermission`-Objekt aufgerufen.

(Der folgende Code entspricht unserem Code aus der ersten Anleitung.)

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

Sie können das gesamte Codebeispiel im `advanced.CreateSecurityGroupApp.java`-Codebeispiel einsehen. Hinweis: Sie müssen diese Anwendung nur einmal ausführen, um eine neue Sicherheitsgruppe zu erstellen.

> **Note**
>
> Sie können die Sicherheitsgruppe auch mithilfe von AWS Toolkit for Eclipse erstellen. Siehe .Verwalten von SicherheitsgruppenAWS Cost ExplorerimAWS Toolkit for Eclipse-Benutzerhandbuch für weitere Informationen:

Detaillierte Optionen für die Erstellung von Spot-Instance-Anforderungen

Wie wir in erklärt habenTutorial:Amazon EC2 Spot-Instanceserstellen Sie Ihre Anforderung mit einem Instance-Typ, einem Amazon Machine Image (AMI) und einem Höchstpreis für Ihr Gebot.

Als Erstes erstellen wir ein `RequestSpotInstanceRequest`-Objekt. Für das Anforderungsobjekt sind die Anzahl der gewünschten Instances sowie der Gebotspreis nötig. Außerdem müssen wir die `LaunchSpecification` für die Anforderung festlegen. Sie umfasst den Instance-Typ, die AMI-ID sowie die Sicherheitsgruppe, die Sie verwenden möchten. Nachdem die Anforderung vorbereitet ist,

rufen wir die Methode `requestSpotInstances` des Objekts `AmazonEC2Client` auf. Es folgt ein Beispiel für die Anforderung einer Spot-Instance.

(Der folgende Code entspricht unserem Code aus der ersten Anleitung.)

```
// 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);
```

Persistente im Vergleich zu einmaligen Anforderungen

Beim Erstellen einer Spot-Instance-Anforderung können Sie mehrere optionale Parameter angeben. Die erste Option gibt an, ob Ihre Anforderung von einmaliger oder persistenter Natur sein soll. Standardmäßig handelt es sich um eine einmalige Anforderung. Eine einmalige Anforderung kann nur einmal erfüllt werden. Sind die angeforderten Instances beendet, wird die Anforderung geschlossen. Eine persistente Anforderung wird zur Erfüllung immer dann herangezogen, wenn für die gleiche

Anforderung keine Spot-Instance ausgeführt wird. Geben Sie den Typ der Anforderung an, indem Sie einfach den "Type" auf der Spot-Anforderung festlegen. Dies lässt sich mit folgendem Code erreichen:

```
// 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);
```

Einschränken der Dauer einer Anforderung

Außerdem können Sie optional festlegen, wie lange Ihre Anforderung gültig bleibt. Sie können für diesen Zeitraum eine Start- und Endzeit festlegen. Standardmäßig wird eine Spot-Anforderung zur Erfüllung von dem Augenblick ihrer Erstellung bis zu dem Zeitpunkt herangezogen, an dem sie entweder erfüllt oder von Ihnen storniert wird. Allerdings können Sie die Gültigkeitsdauer bei Bedarf einschränken. Ein Beispiel dafür, wie Sie diesen Zeitraum angeben, wird im folgenden Code gezeigt:

```
// 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);
```

Gruppieren Ihrer Amazon EC2-Spot-Instance-Anforderungen

Auf Wunsch können Sie Spot-Instance-Anforderungen auf verschiedene Weise gruppieren. Beachten Sie die Vorteile der Nutzung von Start-, Availability Zone- und Platzierungsgruppen.

Wenn Sie sichergehen möchten, dass Ihre Spot-Instances gleichzeitig gestartet und beendet werden, können Sie eine Startgruppe nutzen. Eine Startgruppe ist eine Bezeichnung, die einige Gebote gruppiert. Alle Instances in einer Startgruppe werden zusammen gestartet und beendet. Hinweis: Wurden Instances in einer Startgruppe bereits erfüllt, gibt es keine Garantie dafür, dass neu in der gleichen Startgruppe gestarteten Instances ebenfalls erfüllt werden. Ein Beispiel dafür, wie Sie eine Startgruppe festlegen können, wird im folgenden Code gezeigt:

```
// 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);
```

Wenn Sie sicherstellen möchten, dass alle Instances in einer Anforderung in derselben Availability Zone gestartet werden, und es nicht relevant ist, in welcher Zone, können Sie Availability Zone-Gruppen nutzen. Eine Availability Zone-Gruppe ist eine Bezeichnung, die eine Gruppe von Instances in derselben Availability Zone gruppiert. Alle Instances mit der gleichen Availability Zone-Gruppe, die gleichzeitig erfüllt werden, starten in derselben Availability Zone. Ein Beispiel für die Einrichtung einer Availability Zone-Gruppe finden Sie hier:

```
// 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);
```

Sie können eine Availability Zone angeben, die Sie für Ihre Spot-Instances nutzen möchten. Das folgende Codebeispiel zeigt, wie Sie eine Availability Zone festlegen.

```
// 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);
```

Schließlich können Sie eine Platzierungsgruppe angeben, wenn Sie High Performance Computing (HPC)-Spot-Instances nutzen, etwa Cluster Compute-Instances oder Cluster-GPU-Instances. Platzierungsgruppen sorgen für eine niedrigere Latenz und eine hohe Bandbreitenkonnektivität zwischen den Instances. Ein Beispiel für die Einrichtung einer Platzierungsgruppe finden Sie hier:

```
// 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);
```

Alle Parameter in diesem Abschnitt sind optional. Es ist wichtig zu wissen, dass die meisten dieser Parameter — mit Ausnahme davon, ob Ihr Gebot einmalig oder persistent ist — die Wahrscheinlichkeit der Gebotserfüllung verringern können. Daher ist es wichtig, diese Optionen nur bei Bedarf zu nutzen. Alle obigen Codebeispiele sind in einem langen Codebeispiel kombiniert, das in der `com.amazonaws.codesamples.advanced.InlineGettingStartedCodeSampleApp.java`-Klasse zu finden ist.

So bleibt eine Stammpartition nach einer Unterbrechung oder Beendigung erhalten

Eine der einfachsten Methoden zum Verwalten von Unterbrechungen Ihrer Spot-Instances besteht darin, sicherzustellen, dass Ihre Daten in einem Amazon Elastic Block Store (Amazon Elastic Block Store) als Prüfpunkt gesperrt werdenAmazon EBS) Volumen bei regelmäßiger Trittfrequenz. Durch das Setzen von Prüfpunkten in regelmäßigen Abständen verlieren Sie im Falle einer Unterbrechung nur die seit dem letzten Prüfpunkt erstellten Daten (angenommen, zwischenzeitlich wurden keine anderen idempotenten Aktionen ausgeführt). Um diesen Prozess zu vereinfachen, können Sie Ihre Spot-Anforderung so konfigurieren, dass Ihre Stammpartition bei Unterbrechungen oder Beendigungen nicht gelöscht wird. Im folgenden Beispiel haben wir neuen Code eingefügt, der zeigt, wie sich dieses Szenario umsetzen lässt.

Im hinzugefügten Code erstellen wir ein`BlockDeviceMapping`Objekt und setze sein zugeordnetesAmazon Elastic Block Store(Amazon EBS) zu einemAmazon EBSObjekt, für das wir konfiguriert habennotgelöscht werden, wenn die Spot-Instance beendet wird. Dann fügen wir diese `BlockDeviceMapping` zur ArrayList der Zuordnungen hinzu, die wir in die Start-Spezifikation einschließen.

```
// 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);
```

Wenn Sie dieses Volume beim Startup erneut an Ihre Instance anfügen möchten, können Sie auch die Einstellungen für Blockgerät-Zuweisung verwenden. Wenn Sie eine Nicht-Stammpartition angefügt haben, können Sie alternativ auch Amazon angeben.Amazon EBSVolumes, die Sie nach dem Fortsetzen der Spot-Instance anfügen möchten. Geben Sie dazu einfach eine Snapshot-ID in Ihrem `EbsBlockDevice` und einen alternativen Gerätenamen in Ihren `BlockDeviceMapping`-Objekten an. Durch die Nutzung von Blockgerät-Zuweisungen lässt sich die Instance einfacher starten.

Wenn Sie die Stammpartition verwenden, um Prüfpunkte für Ihre wichtigen Daten anzulegen, können Sie auf diese Weise die Wahrscheinlichkeit der Unterbrechung Ihrer Instances im Griff behalten. Weitere Methoden zum Umgang mit der Wahrscheinlichkeit von Unterbrechungen finden Sie im Video Managing Interruption.

So markieren Sie Spot-Anforderungen und -Instances

Hinzufügen von Tags zuAmazon EC2Ressourcen können die Administration Ihrer Cloud-Infrastruktur vereinfachen. Tags sind ein Typ von Metadaten, der verwendet werden kann, um benutzerfreundliche Namen zu erstellen, die Durchsuchbarkeit zu optimieren und die Koordination zwischen mehreren Benutzern zu verbessern. Sie können Tags auch zur Automatisierung von Skripts und Teilen Ihrer Prozesse nutzen. Um mehr über das Tagging zu lesenAmazon EC2ressourcen:Verwenden von TagsimAmazon EC2-Benutzerhandbuch für Linux-Instances.

Markieren von -Anforderungen

Zum Hinzufügen von Tags zu Ihren Spot-Anforderungen müssen Sie sie markieren, nachdem sie angefordert wurden. Über den Rückgabewert von `requestSpotInstances()` erhalten Sie ein RequestSpotInstancesResult-Objekt, mit dem Sie die Spot-Anforderungs-IDs zur Markierung abrufen können:

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

Sobald Sie die IDs haben, können Sie die Anforderungen markieren.CreateTagsRequestund rufen SieAmazon EC2KundencreateTags()-Methode:

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

Markieren von Instances

Ähnlich wie bei Spot-Anforderungen selbst können Sie eine Instance erst nach ihrer Erstellung markieren. Instances werden erstellt, sobald die Spot-Anforderung erfüllt wurde (d. h., wenn sie nicht mehr den Status offen hat).

Sie können den Status Ihrer Anfragen überprüfen, indem Sie dieAmazon EC2Kundendescribe`SpotInstanceRequests()`-Methode mit einemDescribeSpotInstanceRequestsRequest-Objekt. Das zurückgegebene DescribeSpotInstanceRequestsResult-Objekt enthält eine Liste mit SpotInstanceRequest-Objekten, über die Sie den Status Ihrer Spot-Instance-Anforderungen abfragen und deren Instance-IDs erhalten können, sobald sie nicht mehr den Status offen haben.

Sobald die Spot-Anforderung nicht mehr offen ist, können Sie ihre Instance-ID vom `SpotInstanceRequest`-Objekt erhalten, indem Sie dessen `getInstanceId()`-Methode aufrufen.

```
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);
```

Jetzt können Sie die zurückgegebenen Instances markieren:

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

Stornieren von Spot-Anforderungen und Beenden von Instances

Stornieren einer Spot-Anforderung

Zum Abbrechen einer Spot-Instance-Anforderung rufen Sie`cancelSpotInstanceRequests`auf derAmazon EC2Kunde mit einemCancelSpotInstanceRequestsRequest-Objekt.

```
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());
}
```

Beenden von Spot-Instances

Sie können laufende Spot-Instances beenden, indem Sie ihre IDs an dieAmazon EC2Kunden`terminateInstances()`-Methode.

```
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());
}
```

Zusammenfassung

Fassen wir zusammen: Wir bieten einen eher objektorientierten Ansatz, der die in dieser Anleitung gezeigten Schritte in einer einfach einsetzbaren Klasse kombiniert. Wir instanziieren eine Klasse namens `Requests`, die diese Aktionen ausführt. Zudem legen wir eine Klasse `GettingStarted`App mit einer main-Methode an, in der wir die High-Level-Funktionsaufrufe durchführen.

Den vollständigen Quelltext dieses Beispiels finden Sie zum Lesen und Herunterladen auf GitHub.

Herzlichen Glückwunsch! Sie haben jetzt die Anleitung "Erweiterte Anforderungsfunktionen" zur Entwicklung von Spot-Instance-Software mit dem AWS SDK for Java abgeschlossen.

Verwalten von Amazon EC2-Instances

Erstellen einer Instance

Erstellen eines neuenAmazon EC2Instanz durch Aufruf der Amazonec2Client's`runInstances`Methode, um es mit einemRunInstancesRequestEnthalten Sie dieAmazon Machine Image (AMI)zu benutzen und einInstance-Typaus.

Importe

```
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;
```

Code

```
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();
```

Siehe vollständiges Beispiel.

Starten einer Instance

So starten Sie einAmazon EC2Instanz, rufen Sie die Amazonec2Client's`startInstances`Methode, um es mit einemstartInstancesRequestEnthält die ID der zu startenden -Instance.

Importe

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

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

Code

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

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

ec2.startInstances(request);
```

Siehe vollständiges Beispiel.

Anhalten einer Instance

So halten Sie ein anAmazon EC2Instanz, rufen Sie die Amazonec2Client's`stopInstances`Methode, um es mit einemstopInstancesRequestEnthält die ID der anzuhaltenden -Instance.

Importe

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

Code

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

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

ec2.stopInstances(request);
```

Siehe vollständiges Beispiel.

Neustarten einer -Instance

So starten Sie einAmazon EC2Instanz, rufen Sie die Amazonec2Client's`rebootInstances`Methode, um es mit einemRebootInstancesRequestenthält die ID der zu neustartenden Instanz.

Importe

```
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;
```

Code

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

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

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

Siehe vollständiges Beispiel.

Beschreiben von -Instances

Erstellen Sie zum Auflisten Ihrer Instances eineDescribeInstancesRequestund rufen Sie den Amazonec2Client's and`describeInstances`-Methode. Sie gibt ein DescribeInstancesResult-Objekt zurück, mit dem Sie die Amazon EC2-Instances für Ihr Konto und Ihre Region auflisten können.

Instances werden nach Reservierung gruppiert. Jede Reservierung entspricht dem Aufruf von `startInstances`, durch den die Instance gestartet wurde. Um Ihre Instances aufzulisten, sollten Sie zuerst die `getReservations' method, and then call `getInstances`-Methode der `DescribeInstancesResult`-Klasse für jedes zurückgegebene Reservation-Objekt aufrufen.

Importe

```
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;
```

Code

```
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;
    }
}
```

Die Ergebnisse werden seitenweise zurückgegeben. Sie können die weiteren Ergebnisse abrufen, indem Sie den von der `getNextToken`-Methode des Rückgabeobjekts zurückgegebenen Wert an die `setNextToken`-Methode des Original-Anforderungsobjekts übergeben. Verwenden Sie dann das gleiche Anforderungsobjekt für den nächsten Aufruf von `describeInstances`.

Siehe vollständiges Beispiel.

Überwachung einer Instance

Sie können verschiedene Aspekte Ihrer Amazon EC2-Instances überwachen, wie CPU- und Netzwerkauslastung, verfügbarer Arbeitsspeicher und verbleibender Festplattenspeicher. Weitere Informationen zur Instance-Überwachung finden Sie unterÜberwachungAmazon EC2imAmazon EC2Benutzerhandbuch für Linux-Instances.

Um mit der Überwachung einer Instanz zu beginnen, müssen Sie eineMonitorInstancesRequestMit der ID der zu überwachenden Instance und übergeben Sie sie an die Amazonec2Client'smonitorInstances-Methode.

Importe

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

Code

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

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

ec2.monitorInstances(request);
```

Siehe vollständiges Beispiel.

Anhalten der Instance-Überwachung

Um die Überwachung einer Instanz zu beenden, erstellen Sie eineUnmonitorInstancesRequestMit der ID der Instance, die nicht mehr überwacht werden soll, und übergeben Sie sie an die Amazonec2Client'sunmonitorInstances-Methode.

Importe

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

Code

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

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

ec2.unmonitorInstances(request);
```

Siehe vollständiges Beispiel.

Weitere Informationen

- RunInstancesimAmazon EC2-API-Referenz
- DescribeInstancesimAmazon EC2-API-Referenz
- StartInstancesimAmazon EC2-API-Referenz
- StopInstancesimAmazon EC2-API-Referenz
- RebootInstancesimAmazon EC2-API-Referenz
- MonitorInstancesimAmazon EC2-API-Referenz
- UnmonitorInstancesimAmazon EC2-API-Referenz

Verwenden von Elastic IP-Adressen in Amazon EC2

EC2-Classic geht in den Ruhestand

> **Warning**
>
> EC2-Classic wird am 15. August 2022 eingestellt. Wir empfehlen Ihnen die Migration von EC2-Classic zu einer VPC. Weitere Informationen finden Sie unterMigration von EC2-Classic zu einer VPCimAmazon EC2-Benutzerhandbuch für Linux-Instancesoder dasAmazon EC2 EC2-Benutzerhandbuch für Windows-Instances. Siehe auch den Blog-PostingEC2-Classic-Classic-Networking geht in den Ruhestand — So bereiten Sie sich vor.

Zuweisen einer Elastic IP-Adresse

Um eine Elastic IP-Adresse zu verwenden, verknüpfen Sie sie zuerst mit Ihrem Konto und anschließend mit Ihrer Instance oder Netzwerkschnittstelle.

Sie können eine Elastic IP-Adresse zuordnen, indem Sie eine Elastic IP-Adresse zuordnen`allocateAddress`-Methode mit einemAllocateAddressRequest-Methode, die den Netzwerktyp enthält (klassisches EC2 oder VPC).

den von zurückgegebenenAllocateAddressResultenthält eine Zuordnungs-ID, mit der Sie die Adresse mit einer Instance verknüpfen können. Übergeben Sie dazu die Zuordnungs-ID und die Instance-ID in einemAssociateAddressRequestzu den AmazonEC2Clients`associateAddress`-Methode.

Importe

```
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;
```

Code

```
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);
```

Siehe vollständiges Beispiel.

Beschreiben von Elastic IP-Adressen

Rufen Sie den AmazonEC2-Client auf, um die Elastic IP-Adressen aufzulisten, die Ihrem Konto zugewiesen sinddescribeAddresses-Methode. Sie gibt zurück.DescribeAddressesResultmit der Sie eine Liste von abrufen könnenAdresse-Methode, die die Elastic IP-Adressen in Ihrem Konto darstellen.

Importe

```
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;
```

Code

```
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());
}
```

Siehe vollständiges Beispiel.

Freigeben einer Elastic IP-Adresse

Sie können eine Elastic IP-Adresse freigeben, indem Sie eine Elastic IP-Adresse freigeben`releaseAddress`Methode, übergibt es aReleaseAddressRequestenthält die Zuordnungs-ID der Elastic IP-Adresse, die Sie freigeben möchten.

Importe

```
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;
```

Code

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

ReleaseAddressRequest request = new ReleaseAddressRequest()
```

```
    .withAllocationId(alloc_id);

ReleaseAddressResult response = ec2.releaseAddress(request);
```

Sobald Sie eine Elastic IP-Adresse freigeben, wird sie an denAWSIP-Adresspool und ist möglicherweise danach nicht mehr für Sie verfügbar. Achten Sie darauf, die DNS-Datensätze sowie alle Server und Geräte zu aktualisieren, die mit der Adresse kommunizieren. Beim Versuch, eine Elastic IP-Adresse freizugeben, die Sie bereits freigegeben hatten, bekommen Sie eineAuthFailureFehler wenn die Adresse bereits einer anderen zugewiesen istAWS-Konto.

Wenn Sie EC2-Classic oder eine Standard-VPC verwenden, heben Sie durch das Freigeben einer Elastic IP-Adresse automatisch die Verknüpfung mit allen Instances auf, mit der sie verknüpft war. Rufen Sie die Verknüpfung einer Elastic IP-Adresse auf, ohne sie freizugebendisassociateAddress-Methode.

Wenn Sie einen Nicht-Standard-VPC verwenden, müssen Sie die Verknüpfung der Elastic IP-Adresse mit `disassociateAddress` aufheben, bevor Sie versuchen, sie freizugeben. AnsonstenAmazon EC2gibt einen Fehler zurück (InvalidIPAddress.InUse) enthalten.

Siehe vollständiges Beispiel.

Weitere Informationen

- Elastic IP-AdressenimAmazon EC2Benutzerhandbuch für Linux-Instances
- AllocateAddressimAmazon EC2-API-Referenz
- DescribeAddressesimAmazon EC2-API-Referenz
- ReleaseAddressimAmazon EC2-API-Referenz

Verwenden von Regionen und Availability Zones

Beschreiben von Regionen

Um die für Ihr Konto verfügbaren Regionen aufzulisten, rufen Sie den Amazonec2Clients aufdescribeRegions-Methode. Sie gibt ein DescribeRegionsResult zurück. Rufen Sie die `getRegions`-Methode des zurückgegebenen Objekts auf und Sie erhalten eine Liste mit Region-Objekten, von denen jedes für eine Region steht.

Importe

```
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;
```

Code

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

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

Siehe vollständiges Beispiel.

Beschreiben von Availability Zones

Rufen Sie den Amazonec2Clients an, um die für Ihr Konto verfügbaren Availability Zones aufzulistendescribeAvailabilityZones-Methode. Sie gibt ein DescribeAvailabilityZonesResult zurück. Rufen Sie dessen getAvailabilityZones-Methode auf und Sie erhalten eine Liste mit AvailabilityZone-Objekten, von denen jedes für eine Availability Zone steht.

Importe

```
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;
```

Code

```
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());
}
```

Siehe vollständiges Beispiel.

Beschreiben von Konten

Um Ihr Konto zu beschreiben, rufen Sie den Amazonec2Client's`describeAccountAttributes`-Methode. Diese Methode gibt ein DescribeAccountAttributesResult-Objekt zurück. Rufen Sie diese `getAccountAttributes`-Objektmethode auf, um eine Liste von AccountAttribute-Objekten abzurufen. Sie können die Liste durchlaufen, um ein AccountAttribute-Objekt abzurufen.

Sie können die Attributwerte Ihres Kontos abrufen, indem Sie dieAccountAttribute-Objekte`getAttributeValues`-Methode. Diese Methode gibt eine Liste von AccountAttributeValue-Objekten zurück. Sie können diese zweite Liste durchlaufen, um den Wert von Attributen anzuzeigen (siehe das folgende Codebeispiel).

Importe

```
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;
```

Code

```
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();
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- -Regionen und Availability ZonesimAmazon EC2Benutzerhandbuch für Linux-Instances
- DescribeRegionsimAmazon EC2-API-Referenz
- DescribeAvailabilityZonesimAmazon EC2-API-Referenz

Arbeiten mit Amazon EC2-Schlüsselpaaren

Erstellen eines Schlüsselpaars

Um ein key pair zu erstellen, rufen Sie den Amazonec2Client's auf`createKeyPair`-Methode mit einemCreateKeyPairRequestdas enthält den Namen des Schlüssels.

Importe

```
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;
```

Code

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

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

CreateKeyPairResult response = ec2.createKeyPair(request);
```

Siehe vollständiges Beispiel.

Beschreiben von Schlüsselpaaren

Um Ihre Schlüsselpaare aufzulisten und Informationen über sie zu erhalten, rufen Sie den Amazonec2Client's and`describeKeyPairs`-Methode. Sie gibt ein DescribeKeyPairsResult zurück, mit dem Sie auf die Liste der Schlüsselpaare zugreifen können. Rufen Sie dazu dessen `getKeyPairs`-Methode auf, die eine Liste mit KeyPairInfo-Objekten zurückgibt.

Importe

```
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;
```

Code

```
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());
}
```

Siehe vollständiges Beispiel.

Löschen eines Schlüsselpaars

Um ein key pair zu löschen, rufen Sie den Amazonec2Client's`deleteKeyPair`Methode, übergibt es einDeleteKeyPairRequestdas enthält den Namen des zu löschenden key pair.

Importe

```
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;
```

Code

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

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

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

Siehe vollständiges Beispiel.

Weitere Informationen

- Amazon EC2Schlüsselpaareim Amazon EC2Benutzerhandbuch für Linux-Instances
- CreateKeyPairim Amazon EC2-API-Referenz
- DescribeKeyPairsim Amazon EC2-API-Referenz
- DeleteKeyPairim Amazon EC2-API-Referenz

Arbeiten mit Sicherheitsgruppen in Amazon EC2

Erstellen einer Sicherheitsgruppe

Um eine Sicherheitsgruppe zu erstellen, rufen Sie die`createSecurityGroup` Methode des AmazonEC2Clients mit einem auf CreateSecurityGroupRequest, das den Namen des Schlüssels enthält.

Importe

```
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;
```

Code

```
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);
```

Siehe vollständiges Beispiel.

Konfigurieren einer Sicherheitsgruppe

Eine Sicherheitsgruppe kann eingehenden und ausgehenden Datenverkehr Ihrer Amazon EC2-Instances steuern.

Um Ihrer Sicherheitsgruppe Eingangsregeln hinzuzufügen, verwenden Sie die`authorizeSecurityGroupIngress` Methode von AmazonEC2Client und geben Sie den Namen der Sicherheitsgruppe und die Zugriffsregeln (IpPermission) an, die Sie ihr innerhalb eines AuthorizeSecurityGroupIngressRequestObjekts zuweisen möchten. Im folgenden Beispiel wird gezeigt, wie Sie einer Sicherheitsgruppe IP-Berechtigungen hinzufügen.

Importe

```
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;
```

Code

```
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);
```

Um der Sicherheitsgruppe eine Ausgangsregel hinzuzufügen, geben Sie ähnliche Daten in und wie AuthorizeSecurityGroupEgressRequestdie`authorizeSecurityGroupEgress` Methode von AmazonEC2Client an.

Siehe vollständiges Beispiel.

Beschreiben von Sicherheitsgruppen

Rufen Sie die`describeSecurityGroups` Methode von AmazonEC2Client auf, um Ihre Sicherheitsgruppen zu beschreiben oder Informationen über sie zu erhalten. Es gibt eine zurück DescribeSecurityGroupsResult, mit der Sie auf die Liste der Sicherheitsgruppen zugreifen können, indem Sie ihre`getSecurityGroups` Methode aufrufen, die eine Liste von SecurityGroupObjekten zurückgibt.

Importe

```
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;
```

Code

```
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];
```

Siehe vollständiges Beispiel.

Löschen einer Sicherheitsgruppe

Um eine Sicherheitsgruppe zu löschen, rufen Sie die`deleteSecurityGroup` Methode von AmazonEC2Client auf und übergeben Sie ihr eine DeleteSecurityGroupRequest, die die ID der zu löschenden Sicherheitsgruppe enthält.

Importe

```
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;
```

Code

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

DeleteSecurityGroupRequest request = new DeleteSecurityGroupRequest()
    .withGroupId(group_id);

DeleteSecurityGroupResult response = ec2.deleteSecurityGroup(request);
```

Siehe vollständiges Beispiel.

Weitere Informationen

- Amazon EC2Sicherheitsgruppen imAmazon EC2 Benutzerhandbuch für Linux-Instances
- Autorisierung des eingehenden Datenverkehrs für Ihre Linux-Instances imAmazon EC2 Benutzerhandbuch für Linux-Instances
- CreateSecurityGroupin derAmazon EC2 API-Referenz
- DescribeSecurityGroupsin derAmazon EC2 API-Referenz
- DeleteSecurityGroupin derAmazon EC2 API-Referenz
- AuthorizeSecurityGroupIngressin derAmazon EC2 API-Referenz

IAM-BeispieleAWS SDK for Java

Dieser Abschnitt enthält Beispiele für die Programmierung von IAM mit dem AWS SDK for Java.

AWS Identity and Access ManagementMit (IAM) können Sie den Zugriff aufAWSDienste und Ressourcen für Ihre Benutzer. Mithilfe von IAM können Sie anlegen und verwaltenAWSBenutzer und Gruppen und mittels Berechtigungen ihren Zugriff zulassen oder verweigernAWSRessourcen schätzen. Eine umfassende IAM finden Sie imIAM-Benutzerhandbuchaus.

> **Note**
>
> Die Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der komplette Beispiel-Code steht auf GitHub bereit. Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

Themen

- Verwalten von IAM-Zugriffsschlüsseln
- Verwalten von IAM-Benutzern
- Verwenden von IAM-Konto-Aliasen
- Arbeiten mit IAM-Richtlinien
- Arbeiten mit IAM-Serverzertifikaten

Verwalten von IAM-Zugriffsschlüsseln

Erstellen eines Zugriffsschlüssels

Um einen IAM-Zugriffsschlüssel zu erstellen, rufen Sie den AmazonIdentityManagementClient auf`createAccessKey`-Methode mit einemCreateAccessKeyRequest-Objekt.

`CreateAccessKeyRequest`that zwei Konstruktoren: einen mit Benutzernamen und einen weiteren ohne Parameter. Wenn Sie die Version nutzen, die keine Parameter entgegen nimmt, müssen Sie den Benutzernamen mithilfe der `withUserName`-Setter-Methode festlegen, bevor Sie das Element an die `createAccessKey`-Methode übergeben.

Importe

```
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;
```

Code

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

CreateAccessKeyRequest request = new CreateAccessKeyRequest()
    .withUserName(user);

CreateAccessKeyResult response = iam.createAccessKey(request);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Auflisten von Zugriffsschlüsseln

Erstellen Sie zum Auflisten der Zugriffsschlüssel für einen bestimmten BenutzerListAccessKeysRequest-Objekt, das den Benutzernamen enthält, für den die Schlüssel aufgelistet werden sollen. Übergeben Sie das Objekt an den AmazonIdentityManagementClient`listAccessKeys`-Methode.

> **Note**
>
> Wenn Sie keinen Benutzernamen angeben`listAccessKeys`wird es versuchen, Zugriffsschlüssel aufzulisten, die mit demAWS-Kontodas unterzeichnete den Antrag.

Importe

```
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;
```

Code

```
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;
    }
}
```

Die Ergebnisse von `listAccessKeys` sind seitenweise angeordnet (mit einem Standardhöchstwert von 100 Datensätzen pro Aufruf). Sie können `getIsTruncated` für das zurückgegebene ListAccessKeysResult-Objekt aufrufen, um herauszufinden, ob die Abfrage weniger als die insgesamt verfügbaren Ergebnisse zurückgegeben hat. Falls ja, rufen Sie `setMarker` für den `ListAccessKeysRequest` auf und übergeben Sie ihn beim nächsten Aufruf von `listAccessKeys`.

Auf GitHub finden Sie ein vollständiges Beispiel.

Abrufen der letzten Nutzungszeit eines Zugriffsschlüssels

Um den Zeitpunkt abzufragen, wann ein Zugriffsschlüssel zuletzt verwendet wurde, rufen Sie den AmazonIdentityManagementClient auf`getAccessKeyLastUsed`-Methode mit der -ID des Zugriffsschlüssels (die unter Verwendung eines`GetAccessKeyLastUsedRequest`-Objekt oder direkt zu der Überlastung, die die Zugriffsschlüssel-ID direkt übernimmt.

Anschließend können Sie die letzte Nutzungszeit des Schlüssels mit dem zurückgegebenen GetAccessKeyLastUsedResult-Objekt abfragen.

Importe

```
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;
```

Code

```
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());
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Aktivieren oder Deaktivieren von Zugriffsschlüsseln

Sie können einen Zugriffsschlüssel aktivieren oder deaktivieren, indem Sie eineupdateAccessKeyRequestObjekt, das die Zugriffsschlüssel-ID, optional den Benutzernamen und den gewünschtenStatusübergeben Sie das Anforderungsobjekt an den AmazonIdentityManagementClient`updateAccessKey`-Methode.

Importe

```
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;
```

Code

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

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

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen eines Zugriffsschlüssels

Um einen Zugriffsschlüssel dauerhaft zu löschen, rufen Sie den AmazonIdentityManagementClient's auf`deleteKey`Methode, um es mit einemdeleteAccessKeyRequestenthält die ID und den Benutzernamen des Zugriffsschlüssels.

> **Note**
>
> Nach dem Löschen können Schlüssel nicht mehr abgerufen oder verwendet werden. Wenn Sie einen Schlüssel vorübergehend deaktivieren möchten, sodass er später erneut aktiviert werden kann, verwenden Sie stattdessen die updateAccessKey-Methode.

Importe

```
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;
```

Code

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

DeleteAccessKeyRequest request = new DeleteAccessKeyRequest()
    .withAccessKeyId(access_key)
    .withUserName(username);

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- CreateAccessKeyin der IAM-API-Referenz
- ListAccessKeysin der IAM-API-Referenz
- GetAccessKeyLastUsedin der IAM-API-Referenz
- UpdateAccessKeyin der IAM-API-Referenz
- DeleteAccessKeyin der IAM-API-Referenz

Verwalten von IAM-Benutzern

Erstellen eines Benutzers

Erstellen Sie einen neuen IAM-Benutzer, indem Sie den Benutzernamen für dieAmazonIdentityManagementClient`createUser` 's-Methode angeben, entweder direkt oder mithilfe eines CreateUserRequestObjekts, das den Benutzernamen enthält.

Importe

```
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;
```

Code

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

CreateUserRequest request = new CreateUserRequest()
    .withUserName(username);

CreateUserResult response = iam.createUser(request);
```

Das vollständige Beispiel finden Sie aufGitHub.

Auflisten von Benutzern

Um die IAM-Benutzer für Ihr Konto aufzulisten, erstellen Sie ein neues Konto ListUsersRequestund übergeben Sie es an dieAmazonIdentityManagementClient`listUsers` Methode. Sie können die Liste der Benutzer abrufen, indem Sie das zurückgegebene ListUsersResultObjekt aufrufen`getUsers`.

Die von `listUsers` zurückgegebene Benutzerliste ist segmentiert. Sie können prüfen, ob weitere Ergebnisse bereitliegen, indem Sie die `getIsTruncated`-Methode des Antwortobjekts aufrufen. Gibt sie `true` zurück, rufen Sie die `setMarker()`-Methode des Anfrageobjekts auf und übergeben ihr den Rückgabewert der `getMarker()`-Methode des Antwortobjekts.

Importe

```
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;
```

Code

```
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;
    }
}
```

Das vollständige Beispiel finden Sie aufGitHub.

Aktualisieren eines Benutzers

Um einen Benutzer zu aktualisieren, rufen Sie die`updateUser` Methode desAmazonIdentityManagementClient Objekts auf, die ein UpdateUserRequestObjekt verwendet, mit dem Sie den Namen oder Pfad des Benutzers ändern können.

Importe

```
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;
```

Code

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

UpdateUserRequest request = new UpdateUserRequest()
    .withUserName(cur_name)
    .withNewUserName(new_name);

UpdateUserResult response = iam.updateUser(request);
```

Das vollständige Beispiel finden Sie aufGitHub.

Löschen eines Benutzers

Um einen Benutzer zu löschen, rufen SieAmazonIdentityManagementClient die`deleteUser` Anfrage mit einem UpdateUserRequestObjektsatz mit dem zu löschenden Benutzernamen auf.

Importe

```
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;
```

Code

```
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;
}
```

Das vollständige Beispiel finden Sie aufGitHub.

Weitere Informationen

- IAM-Benutzer imIAM Benutzerhandbuch
- Verwaltung von IAM-Benutzern imIAM Benutzerhandbuch
- CreateUserin der IAM-API-Referenz
- ListUsersin der IAM-API-Referenz
- UpdateUserin der IAM-API-Referenz
- DeleteUserin der IAM-API-Referenz

Verwenden von IAM-Konto-Aliasen

Wenn die URL der Anmeldeseite den Namen des Unternehmens oder andere benutzerfreundliche Kennungen anstelle derAWS-Konto -ID enthalten soll, können Sie einen Alias für das Konto erstellenAWS-Konto.

> **Note**
>
> AWSunterstützt genau einen Kontoalias pro Konto.

Erstellen eines Konto-Alias

Um einen Kontoalias zu erstellen, rufen Sie die`createAccountAlias` MethodeAmazonIdentityManagementClient's mit einem CreateAccountAliasRequestObjekt auf, das den Aliasnamen enthält.

Importe

```
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;
```

Code

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

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

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

Das vollständige Beispiel finden Sie aufGitHub.

Auflisten von Konto-Aliassen

Listen Sie Ihre Konto-Aliasse auf, falls vorhanden, indem Sie die `listAccountAliases`-Methode des AmazonIdentityManagementClient aufrufen.

> **Note**
>
> Die zurückgegebene Methode ListAccountAliasesResultunterstützt dieselbengetIsTruncatedgetMarker AND-Methoden wie andereAWS SDK for Java Listenmethoden,AWS-Konto kann jedoch nur einen Kontoalias haben.

Importe

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

Code

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

ListAccountAliasesResult response = iam.listAccountAliases();

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

das vollständige Beispiel finden Sie aufGitHub.

Löschen eines Konto-Alias

Zum Löschen Ihres Konto-Alias rufen Sie die `deleteAccountAlias`-Methode des AmazonIdentityManagementClient auf. Wenn Sie einen Kontoalias löschen, müssen Sie seinen Namen mithilfe eines DeleteAccountAliasRequestObjekts angeben.

Importe

```
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;
```

Code

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

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

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

Das vollständige Beispiel finden Sie aufGitHub.

Weitere Informationen

- IhreAWS Konto-ID und ihr Alias imIAM Benutzerhandbuch
- CreateAccountAliasin der IAM-API-Referenz
- ListAccountAliasesin der IAM-API-Referenz
- DeleteAccountAliasin der IAM-API-Referenz

Arbeiten mit IAM-Richtlinien

Erstellen einer Richtlinie

Geben Sie zum Erstellen einer neuen Richtlinie den Namen sowie ein JSON-formatiertes Richtliniendokument in einemCreatePolicyRequestzu den AmazonIdentityManagementClient'`createPolicy`-Methode.

Importe

```
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;
```

Code

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

CreatePolicyRequest request = new CreatePolicyRequest()
    .withPolicyName(policy_name)
```

```
    .withPolicyDocument(POLICY_DOCUMENT);

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

IAM-Richtlinien sind JSON-Zeichenfolgen mit einemgut dokumentierte Syntaxaus. Hier finden Sie ein Beispiel, das den Zugriff für bestimmte Anfragen an DynamoDB gewährt.

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Abrufen einer Richtlinie

Um eine vorhandene Richtlinie abzurufen, rufen Sie den AmazonIdentityManagementClient's`getPolicy`-Methode zur Verfügung stellt, indem Sie den ARN der Richtlinie innerhalb einesGetPolicyRequest-Objekt.

Importe

```
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;
```

Code

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

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

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Anfügen einer Rollenrichtlinie

Sie können eine Richtlinie an eine IAM anhängen http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html[role], indem Sie den AmazonIdentityManagementClient's aufrufen`attachRolePolicy`-Methode zur Verfügung stellt, indem Sie den Rollenname und den Richtlinien-ARN in einerAttachRolePolicyRequestaus.

Importe

```
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;
```

Code

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

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

iam.attachRolePolicy(attach_request);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Auflisten angefügter Rollenrichtlinien

Durch Aufrufen des AmazonIdentityManagementClients können Sie die angefügten Richtlinien einer Rolle auflisten`listAttachedRolePolicies`-Methode. Sie nimmt ein ListAttachedRolePoliciesRequest-Objekt mit dem Namen der Rolle entgegen, für die die Richtlinien aufgelistet werden sollen.

Rufen Sie `getAttachedPolicies` für das zurückgegebene ListAttachedRolePoliciesResult-Objekt auf, um die Liste der angefügten Richtlinien abzurufen. Die Ergebnisse sind evtl. gekürzt. Gibt die `ListAttachedRolePoliciesResult`-Methode des `getIsTruncated`-Objekts `true` zurück, rufen Sie die `ListAttachedRolePoliciesRequest`-Methode des `setMarker`-Objekts auf. Verwenden Sie das Ergebnis dann in einem weiteren Aufruf von `listAttachedRolePolicies`, um das nächste Teilergebnis abzurufen.

Importe

```
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;
```

Code

```
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());
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Trennen einer Rollenrichtlinie

Zum Trennen einer Richtlinie von einer Rolle rufen Sie den AmazonIdentityManagementClient auf`detachRolePolicy`-Methode zur Verfügung stellt, indem Sie den Rollenname und den Richtlinien-ARN in einemDetachRolePolicyRequestaus.

Importe

```
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;
```

Code

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

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

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- Übersicht über IAM-RichtlinienimIAM-Benutzerhandbuch.
- AWSIAM-RichtlinienreferenzimIAM-Benutzerhandbuch.
- CreatePolicyin der IAM-API Referenz
- GetPolicyin der IAM-API Referenz
- AttachRolePolicyin der IAM-API Referenz
- ListAttachedRolePoliciesin der IAM-API Referenz
- DetachRolePolicyin der IAM-API Referenz

Arbeiten mit IAM-Serverzertifikaten

So aktivieren Sie HTTPS-Verbindungen zu Ihrer Website oder AnwendungAWSbenötigen Sie eine SSL/TLSServerzertifikataus. Sie können ein Serverzertifikat verwenden, das vonAWSCertificate Manager oder einen, den Sie von einem externen Anbieter erhalten haben.

Wir empfehlen, dass Sie Ihre Serverzertifikate mithilfe von ACM bereitstellen, verwalten und bereitstellen. Mit ACM können Sie ein Zertifikat anfordern und es auf IhremAWS-Ressourcen und lassen Sie ACM Zertifikatserneuerungen durchführen. Zertifikate von ACM sind kostenlos. Weitere Informationen zu ACM finden Sie imACM-Benutzerhandbuchaus.

Abrufen eines Serverzertifikats

Sie können ein Serverzertifikat abrufen, indem Sie den AmazonIdentityManagementClient's aufrufen`getServerCertificate`Methode, übergeben sie einGetServerCertificateRequestmit dem Namen des Zertifikats.

Importe

```
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;
```

Code

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

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

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Auflisten von Serverzertifikaten

Sie können Ihre Serverzertifikate auflisten, indem Sie den AmazonIdentityManagementClient's`listServerCertificates`-Methode mit einerListServerCertificatesRequestaus. Sie gibt ein ListServerCertificatesResult zurück.

Rufen Sie die `getServerCertificateMetadataList`-Methode des zurückgegebenen `ListServerCertificateResult`-Objekts auf und Sie erhalten eine Liste mit ServerCertificateMetadata-Objekten, mit denen Sie Sie Informationen über die einzelnen Zertifikate anfordern können.

Die Ergebnisse sind evtl. gekürzt. Gibt die `ListServerCertificateResult`-Methode des `getIsTruncated`-Objekts `true` zurück, rufen Sie die `ListServerCertificatesRequest`-Methode des `setMarker`-Objekts auf. Verwenden Sie das Ergebnis dann in einem weiteren Aufruf von `listServerCertificates`, um das nächste Teilergebnis abzurufen.

Importe

```
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;
```

Code

```
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;
    }
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Aktualisieren eines Serverzertifikats

Sie können den Namen oder den Pfad eines Serverzertifikats aktualisieren, indem Sie den AmazonIdentityManagementClient aufrufen`updateServerCertificate`-Methode. Sie nimmt ein UpdateServerCertificateRequest-Objekt mit dem aktuellen Namen des Zertifikats und entweder einem neuen Namen oder einem neu zu verwendenden Pfad entgegen.

Importe

```
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;
```

Code

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

UpdateServerCertificateRequest request =
    new UpdateServerCertificateRequest()
        .withServerCertificateName(cur_name)
```

```
        .withNewServerCertificateName(new_name);

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen eines Serverzertifikats

Sie können ein Serverzertifikat löschen, indem Sie die des des AmazonIdentityManagementClients aufrufen`deleteServerCertificate`-Methode mit einerDeleteServerCertificateRequestenthält den Namen des Zertifikats.

Importe

```
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;
```

Code

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

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

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- Arbeiten mit Serverzertifikatenim IAM-Benutzerhandbuch
- GetServerCertificatein der IAM-Referenz
- ListServerCertificatesin der IAM-Referenz
- UpdateServerCertificatein der IAM-Referenz

- DeleteServerCertificatein der IAM-Referenz
- ACM-Benutzerhandbuch

Lambda-Beispiele unter Verwenden der AWS SDK for Java

Dieser Abschnitt bietet Beispiele für die Programmierung von Lambda mithilfe des AWS SDK for Java.

> **Note**
>
> Die Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der komplette Beispiel-Code steht auf GitHub bereit. Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

Themen

- Aufrufen, Auflisten und LöschenLambdaFunktionen

Aufrufen, Auflisten und LöschenLambdaFunktionen

Dieser Abschnitt bietet Beispiele für die Programmierung vonLambdaService-Client unter Verwendung desAWS SDK for Javaaus. So erstellen Sie einLambda-Funktion, siheSo erstellen SieAWS LambdaFunktionenaus.

Themen

- Aufruf einer -Funktion
- Auflisten von -Funktionen
- Löschen einer -Funktion

Aufruf einer -Funktion

Sie können einLambdaFunktion durch Erstellen einesAWSLambdaObjekt und Aufruf`invoke`-Methode. Erstellen Sie ein InvokeRequest-Objekt, um zusätzliche Informationen wie den Funktionsnamen und die Nutzlast anzugeben, die an die Lambda-Funktion übergeben werden sollen.

Funktionsnamen erscheinen alsarn:aws:lambda:us-east- 1:5556330391:function:HelloFunctionaus. Sie können den Wert abrufen, indem Sie sich die Funktion imAWS Management Consoleaus.

Um Nutzlastdaten an eine Funktion zu übergeben, rufen SieInvokeRequest-Objekte`withPayload`Method und geben Sie einen String im JSON-Format an, wie im folgenden Codebeispiel gezeigt.

Importe

```
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;
```

Code

Das folgende Codebeispiel zeigt den Aufruf einer Lambda-Funktion.

```
        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());
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Auflisten von -Funktionen

Erstellen Sie ein AWSLambda-Objekt und rufen Sie seine `listFunctions`-Methode auf. Diese Methode gibt ein ListFunctionsResult-Objekt zurück. Sie können die `getFunctions`-Methode dieses Objekts aufrufen, um eine Liste von FunctionConfiguration-Objekten zurückzugeben. Sie können die Liste durchlaufen, um Informationen über die Funktionen abzurufen. Das folgende Java-Codebeispiel zeigt beispielsweise, wie die einzelnen Funktionsnamen abgerufen werden.

Importe

```
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;
```

Code

Das folgende Java-Codebeispiel veranschaulicht, wie eine Liste vonLambda-Funktionsnamen.

```
        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);
        }
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen einer -Funktion

Erstellen Sie ein AWSLambda-Objekt und rufen Sie seine `deleteFunction`-Methode auf. Erstellen Sie ein DeleteFunctionRequest-Objekt und übergeben Sie es an die `deleteFunction`-Methode. Dieses Objekt enthält Informationen wie den Namen der zu löschenden Funktion. Funktionsnamen erscheinen alsarn:aws:lambda:us-east- 1:5556330391:function:HelloFunctionaus. Sie können den Wert abrufen, indem Sie sich die Funktion imAWS Management Consoleaus.

Importe

```
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;
```

Code

Der folgende Java-Code veranschaulicht, wie einLambdaFunktion.

```
        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);
        }
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Amazon Pinpoint-Beispiele unter Verwenden der AWS SDK for Java

Dieser Abschnitt bietet Beispiele für die Programmierung von Amazon Pinpoint mithilfe des AWS SDK for Java.

> **Note**
>
> Die Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der komplette Beispiel-Code steht auf GitHub bereit. Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

Themen

- Erstellen und Löschen von Apps inAmazon Pinpoint
- Erstellen von -Endpunkten inAmazon Pinpoint
- Erstellen von Segmenten inAmazon Pinpoint
- Kampagnen erstellen inAmazon Pinpoint
- Channel-Aktualisierung inAmazon Pinpoint

Erstellen und Löschen von Apps inAmazon Pinpoint

Eine App ist ein Amazon Pinpoint-Projekt, bei dem Sie die Zielgruppe für eine eigenständige Anwendung definieren und die Zielgruppe mit maßgeschneiderten Nachrichten an die App binden. Die Beispiele auf dieser Seite zeigen, wie Sie eine neue App erstellen oder eine bestehende löschen.

Erstellen einer Anwendung

Erstellen Sie eine neue App inAmazon Pinpointindem Sie einen App-Namen für dieCreateAppRequest-Objekt, und übergeben Sie dieses Objekt dann an die AmazonPinPointClients`createApp`-Methode.

Importe

```
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;
```

Code

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

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen einer APP

Um eine App zu löschen, rufen Sie den AmazonPinPointClient's`deleteApp`request mit einemdeleteAppRequest-Objekt, das den zu löschenden App-Namen enthält.

Importe

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

Code

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

pinpoint.deleteApp(deleteRequest);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- -AppsimAmazon Pinpoint-API-Referenz
- AppimAmazon Pinpoint-API-Referenz

Erstellen von -Endpunkten inAmazon Pinpoint

Ein Endpunkt kennzeichnet auf eindeutige Weise ein Benutzergerät, an das Sie mit Amazon Pinpoint Push-Benachrichtigungen senden können. Wenn Ihre App über einen Amazon Pinpoint-Support verfügt, registriert sie automatisch einen Endpunkt mit Amazon Pinpoint, wenn ein neuer Benutzer die App öffnet. Das folgende Beispiel zeigt, wie Sie programmgesteuert einen neuen Endpunkt hinzufügen.

Erstellen eines Endpunkts

Erstellen Sie einen neuen Endpunkt in Amazon Pinpoint, indem Sie die Endpunktdaten in einem EndpointRequest-Objekt bereitstellen.

Importe

```
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;
```

Code

```
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);
```

Dann erstelle einUpdateEndpointRequest-Objekt damit EndpointRequest - Objekt. Bestehen Sie abschließend die UpdateEndpointRequest Objekt gegen die AmazonPinPointClient's`updateEndpoint`-Methode.

Code

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

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- Hinzufügen von EndpunktenimAmazon PinpointEntwicklerhandbuch
- -EndpunktimAmazon Pinpoint-API-Referenz

Erstellen von Segmenten inAmazon Pinpoint

Ein Benutzersegment stellt einen Teil Ihrer Benutzer basierend auf gemeinsamen Merkmalen dar, z. B. Zeitpunkt der letzten App-Öffnung durch einen Benutzer oder verwendetes Gerät. Das folgende Beispiel zeigt, wie ein Benutzersegment definiert wird.

Erstellen eines Segments

Erstellen Sie ein neues Segment in Amazon Pinpoint, indem Sie Dimensionen des Segments in einem SegmentDimensions-Objekt definieren.

Importe

```
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;
```

Code

```
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);
```

Legen Sie dann das SegmentDimensions-Objekt in einer WriteSegmentRequest fest, die verwendet wird, um ein CreateSegmentRequest-Objekt zu erstellen. Dann übermitteln Sie das CreateSegmentRequest Objekt gegen die AmazonPinPointClient'`createSegment`-Methode.

Code

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

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

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- Amazon PinpointSegmenteimAmazon Pinpoint-Benutzerhandbuch
- Erstellen von SegmentenimAmazon PinpointEntwicklerhandbuch
- SegmenteimAmazon Pinpoint-API-Referenz
- SegmentimAmazon Pinpoint-API-Referenz

Kampagnen erstellen inAmazon Pinpoint

Mit diesen Kampagnen können Sie die Bindung zwischen Ihrer App und den Benutzern erhöhen. Sie können eine Kampagne erstellen, um für ein bestimmtes Benutzersegment maßgeschneiderte Nachrichten oder besondere Werbeaktionen bereitzustellen. In diesem Beispiel wird gezeigt, wie eine neue Standard-Kampagne erstellt wird, bei der eine benutzerdefinierte Push-Benachrichtigung an ein bestimmtes Benutzersegment gesendet wird.

Erstellen einer Kampagne

Bevor Sie eine neue Kampagne erstellen, müssen Sie einen Zeitplan und eine Nachricht definieren und diese Werte in einem WriteCampaignRequestObjekt festlegen.

Importe

```
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;
```

Code

```
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);
```

Erstellen Sie dann eine neue Kampagne,Amazon Pinpoint indem Sie sie WriteCampaignRequestmit der Kampagnenkonfiguration einem CreateCampaignRequestObjekt zur Verfügung stellen. Übergeben Sie dasCreateCampaignRequest Objekt abschließend an die`createCampaign` MethodeAmazonPinpointClient's.

Code

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

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

Das vollständige Beispiel finden Sie aufGitHub.

Weitere Informationen

- Amazon PinpointKampagnen imAmazon Pinpoint Benutzerhandbuch

- Kampagnen imAmazon Pinpoint Entwicklerhandbuch erstellen
- Kampagnen in derAmazon Pinpoint API-Referenz
- Kampagne in derAmazon Pinpoint API-Referenz
- Kampagnenaktivitäten in derAmazon Pinpoint API-Referenz
- Kampagnenversionen in derAmazon Pinpoint API-Referenz
- Kampagnenversion in derAmazon Pinpoint API-Referenz

Channel-Aktualisierung inAmazon Pinpoint

Ein Channel definiert die Arten von Plattformen, an die Sie Nachrichten übermitteln können. Das folgende Beispiel zeigt, wie Sie mit dem APNs-Channel eine Nachricht senden.

Aktualisieren eines Channels

Aktivieren Sie einen Channel in Amazon Pinpoint, indem Sie eine App-ID und ein Anforderungsobjekt des Channel-Typs bereitstellen, den Sie aktualisieren möchten. Bei diesem Beispiel wird der APNs-Channel aktualisiert, der das APNSChannelRequest-Objekt benötigt. Stellen Sie diese imUpdateApnsChannelRequestund übergeben Sie das Objekt an die AmazonPinPointClient's`updateApnsChannel`-Methode.

Importe

```
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;
```

Code

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

UpdateApnsChannelRequest updateRequest = new UpdateApnsChannelRequest()
  .withAPNSChannelRequest(request)
```

```
  .withApplicationId(appId);
UpdateApnsChannelResult result = client.updateApnsChannel(updateRequest);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- Amazon PinpointKanäleimAmazon Pinpoint-Benutzerhandbuch
- ADM-KanalimAmazon Pinpoint-API-Referenz
- APNs-KanalimAmazon Pinpoint-API-Referenz
- APNs-Sandbox-ChannelimAmazon Pinpoint-API-Referenz
- APNs-VoIP-KanalimAmazon Pinpoint-API-Referenz
- APNs-VoIP-Sandbox-ChannelimAmazon Pinpoint-API-Referenz
- Baidu-KanalimAmazon Pinpoint-API-Referenz
- E-Mail-Kanal inimAmazon Pinpoint-API-Referenz
- GCM-KanalimAmazon Pinpoint-API-Referenz
- SMS-KanalimAmazon Pinpoint-API-Referenz

Amazon S3-Beispiele unter Verwenden der AWS SDK for Java

Dieser Abschnitt bietet Beispiele für die Programmierung von Amazon S3 mithilfe des AWS SDK for Java.

> **Note**
>
> Die Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der komplette Beispiel-Code steht auf GitHub bereit. Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

Themen

- Erstellen, Auflisten und LöschenAmazon S3Buckets
- Operationen an Amazon S3 Objekten ausführen
- VerwaltenAmazon S3Zugriffsberechtigungen für Buckets und Objekte

- Verwalten des Zugriffs aufAmazon S3Mit Buckets mithilfe von Bucket-Richtlinien
- benutzen TransferManager zumAmazon S3Operationen
- Konfigurieren einesAmazon S3Bucket als Website
- Verwenden vonAmazon S3Clientseitige -Verschlüsselung

Erstellen, Auflisten und LöschenAmazon S3Buckets

Alle Objekte (Dateien) in Amazon S3 müssen in einem Bucket gespeichert werden. Dieser stellt eine Sammlung von Objekten (Container) dar. Jeder Bucket ist mit einem Schlüssel (Namen) bekannt, der eindeutig sein muss. Ausführliche Informationen zu Buckets und deren Konfiguration finden Sie unterArbeiten mitAmazon S3BucketsimAmazon Simple Storage Service-Benutzerhandbuch.

> **Note**
>
> Bewährte Methode
> Wir empfehlen, dass Sie die Lebenszyklus-Regel AbortIncompleteMultipartUpload für Ihre Amazon S3-Buckets aktivieren.
> Diese Regel weist Amazon S3 an, mehrteilige Uploads abzubrechen, die nicht innerhalb einer bestimmten Anzahl von Tagen nach dem Start abgeschlossen werden. Wenn die festgelegte Höchstdauer überschritten wird, bricht Amazon S3 den Upload ab und löscht dann die unvollständigen Upload-Daten.
> Weitere Informationen finden Sie unterLebenszykluskonfiguration für einen Bucket mit VersioningimAmazon S3-Benutzerhandbuch.

> **Note**
>
> Diese Codebeispiele gehen davon aus, dass Sie das Material inVerwendung derAWS SDK for Javaund haben default konfiguriertAWSAnmeldeinformationen, die die Informationen in verwendenEinrichtenAWSAnmeldeinformationen und Region für die Entwicklungaus.

Bucket erstellen

Verwenden Sie die AmazonS3--Clients`createBucket`-Methode. Der neue Bucket wird zurückgegeben. Die `createBucket`-Methode löst eine Ausnahme aus, falls der Bucket bereits vorhanden ist.

> **Note**
>
> Bevor Sie versuchen, einen Bucket zu erstellen, sollten Sie die `doesBucketExist`-Methode aufrufen, um zu prüfen, ob ein gleichnamiger Bucket bereits existiert. Falls ja, wird `true` zurückgegeben, andernfalls `false`.

Importe

```
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;
```

Code

```
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;
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Auflisten von Buckets

Verwenden Sie die AmazonS3--Clients`listBucket`-Methode. Wenn diese Aktion erfolgreich ist, wird eine Liste mit Bucket-Objekten zurückgegeben.

Importe

```
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;
```

Code

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Bucket löschen

Bevor Sie einen Amazon S3-Bucket löschen können, müssen Sie sicherstellen, dass der Bucket leer ist. Andernfalls erhalten Sie eine Fehlermeldung. Wenn Sie einen versionierten Bucket nutzen, müssen Sie außerdem alle versionierten Objekte löschen, die mit dem Bucket verknüpft sind.

> **Note**
>
> Das vollständige Beispiel enthält die einzelnen Schritte in der richtigen Reihenfolge und zeigt eine umfassende Lösung zum Löschen eines Amazon S3-Buckets samt Inhalt.

Themen

- Entfernen von Objekten aus einem nicht versionierten Bucket vor dem Löschen
- Entfernen von Objekten aus einem versionierten Bucket vor dem Löschen
- Löschen eines leeren Buckets

Entfernen von Objekten aus einem nicht versionierten Bucket vor dem Löschen

Verwenden Sie die AmazonS3--Clients`listObjects`-Methode, um die Liste der Objekte abzurufen und`deleteObject`um jeden zu löschen.

Importe

```
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;
```

Code

```
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;
    }
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Entfernen von Objekten aus einem versionierten Bucket vor dem Löschen

Wenn Sie einen versionierten Bucket nutzen, müssen Sie auch alle gespeicherten Versionen der Objekte im Bucket entfernen, bevor der Bucket gelöscht werden kann.

Mit einem Vorgehen ähnlich wie beim Entfernen von Objekten in einem Bucket entfernen Sie versionierte Objekte, indem Sie den AmazonS3--Clients verwenden`listVersions`Methode, um versionierte Objekte aufzulisten, und dann`deleteVersion`um jeden zu löschen.

Importe

```
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;
```

Code

```
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;
    }
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen eines leeren Buckets

Nachdem Sie die Objekte aus einem Bucket (einschließlich etwaiger versionierter Objekte) gelöscht haben, können Sie durch Aufruf des AmazonS3--Clients den Bucket selbst löschen`deleteBucket`-Methode.

Importe

```
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;
```

Code

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Operationen an Amazon S3 Objekten ausführen

Ein Amazon S3-Objekt stellt eine Datei oder eine Sammlung von Daten dar. Jedes Objekt muss in einem Bucket enthalten sein.

> **Note**
>
> Bei diesen Codebeispielen wird vorausgesetzt, dass Sie das Material unter Verwenden der verstehen AWS SDK for Java und die AWS Standardanmeldeinformationen anhand der Informationen unter AWSAnmeldeinformationen einrichten und Region für Entwicklung konfiguriert haben.

Themen

- Hochladen eines Objekts
- Auflisten von Objekten
- Herunterladen eines Objekts
- Kopieren, Verschieben oder Umbenennen von Objekten
- Objekte löschen
- Löschen mehrerer Objekte auf einmal

Hochladen eines Objekts

Verwenden Sie die `putObject` Methode des AmazonS3-Clients und geben Sie einen Bucket-Namen, einen Schlüsselnamen und eine Datei zum Hochladen an. Der Bucket muss vorhanden sein, andernfalls tritt ein Fehler auf.

Importe

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

Code

```
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);
}
```

Das vollständige Beispiel finden Sie unterGitHub.

Auflisten von Objekten

Um eine Liste von Objekten in einem Bucket abzurufen, verwenden Sie die `listObjects` Methode des AmazonS3-Clients und geben Sie den Namen eines Buckets an.

Die `listObjects` Methode gibt ein ObjectListingObjekt zurück, das Informationen über die Objekte im Bucket bereitstellt. Um die Objektnamen (Schlüssel) aufzulisten, verwenden Sie die `getObjectSummaries` Methode, um eine Liste von ObjectSummaryS3-Objekten abzurufen, von denen jedes ein einzelnes Objekt im Bucket darstellt. Rufen Sie dann dessen `getKey`-Methode zum Abrufen des Objektnamens auf.

Importe

```
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;
```

Code

```
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());
}
```

Das vollständige Beispiel finden Sie unterGitHub.

Herunterladen eines Objekts

Verwenden Sie die `getObject` Methode des AmazonS3-Clients und übergeben Sie ihr den Namen eines Buckets und eines Objekts zum Herunterladen. Bei Erfolg gibt die Methode ein S3Object zurück. Der angegebene Bucket und der Objektschlüssel müssen vorhanden sein, andernfalls tritt ein Fehler auf.

Sie können den Inhalt des Objekts anfordern, indem Sie `getObjectContent` für das Objekt `S3Object` aufrufen. Dies gibt ein S3 zurückObjectInputStream, das sich wie ein `InputStream` Standard-Java-Objekt verhält.

Das folgende Beispiel lädt ein Objekt von S3 herunter und speichert die Inhalte in einer Datei mit dem Namen des Objektschlüssels.

Importe

```
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;
```

Code

```
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);
}
```

Das vollständige Beispiel finden Sie unterGitHub.

Kopieren, Verschieben oder Umbenennen von Objekten

Sie können ein Objekt von einem Bucket in einen anderen kopieren, indem Sie die -Methode des AmazonS3-Clients verwenden. `copyObject` Sie nimmt den Namen des Buckets, aus dem kopiert werden soll, das zu kopierende Objekt sowie den Namen des Zielbuckets entgegen.

Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Code

```
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!");
```

Das vollständige Beispiel finden Sie unterGitHub.

> **Note**
>
> Sie können `copyObject` mit deleteObject verwenden, um ein Objekt zu verschieben oder umzubenennen. Kopieren Sie das Objekt dazu als Erstes auf einen neuen Namen (Sie können den gleichen Bucket als Quelle und Ziel angeben) und löschen Sie das Objekt dann von seinem bisherigen Speicherort.

Objekte löschen

Verwenden Sie die `deleteObject` Methode des AmazonS3-Clients und übergeben Sie ihr den Namen eines Buckets und eines Objekts, das gelöscht werden soll. Der angegebene Bucket und der Objektschlüssel müssen vorhanden sein, andernfalls tritt ein Fehler auf.

Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Code

```
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);
}
```

Das vollständige Beispiel finden Sie unterGitHub.

Löschen mehrerer Objekte auf einmal

Mit der `deleteObjects` Methode des AmazonS3-Clients können Sie mehrere Objekte aus demselben Bucket löschen, indem Sie ihre Namen an die Methode link: sdk-for-java DeleteObjectsRequest /v1/reference/com/amazonaws/services/s3/model/ .html übergeben.

Importe

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

Code

```
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);
}
```

Das vollständige Beispiel finden Sie unterGitHub.

VerwaltenAmazon S3Zugriffsberechtigungen für Buckets und Objekte

Sie können Zugriffskontrolllisten (Access Control Lists, ACLs) für Amazon S3-Buckets und Objekte zur präzisen Kontrolle über Ihre Amazon S3-Ressourcen nutzen.

> **Note**
>
> Diese Codebeispiele gehen davon aus, dass Sie das Material in verstehenVerwendung derAWS SDK for Javaund haben default konfiguriertAWSAnmeldeinformationen, die die Informationen in verwendenEinrichtenAWSAnmeldeinformationen und Region für die Entwicklungaus.

Abrufen der Zugriffskontrollliste für einen Bucket

Rufen Sie zum Abrufen der aktuellen ACL für einen Bucket die AmazonS3'`sgetBucketAcl`Methode, übergeben sie dieBucket-Nameabzufragenden. Diese Methode gibt ein AccessControlList-Objekt zurück. Um jede Zugriffsberechtigung in der Liste abzurufen, rufen Sie die `getGrantsAsList`-Methode des Objekts auf. Sie erhalten dann eine Standard-Java-Liste mit Grant-Objekten.

Importe

```
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;
```

Code

```
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);
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Festlegen der Zugriffskontrollliste für einen Bucket

Rufen Sie zum Hinzufügen oder Ändern von Berechtigungen in einer ACL für einen Bucket die AmazonS3'ssetBucketAcl-Methode. Sie nimmt ein AccessControlList-Objekt entgegen, das eine Liste mit Berechtigten und die festzulegenden Zugriffsebenen enthält.

Importe

```
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;
```

Code

```
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**
>
> Sie können den eindeutigen Bezeichner des Berechtigten direkt mit der Grantee-Klasse angeben. Verwenden Sie alternativ die EmailAddressGrantee-Klasse, um den Berechtigten per E-Mail festzulegen, wie wir dies hier getan haben.

Auf GitHub finden Sie ein vollständiges Beispiel.

Abrufen der Zugriffskontrollliste für ein Objekt

Rufen Sie zum Abrufen der aktuellen ACL für ein Objekt die AmazonS3's`getObjectAcl`Methode, übergeben sie dieBucket-NameundObjektnameabzufragenden. Wie `getBucketAcl` gibt diese Methode ein AccessControlList-Objekt zurück, mit dem Sie jeden Grant untersuchen können.

Importe

```
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;
```

Code

```
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);
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Festlegen der Zugriffskontrollliste für ein Objekt

Rufen Sie zum Hinzufügen oder Ändern von Berechtigungen in einer ACL für ein Objekt die AmazonS3'ssetObjectAcl-Methode. Sie nimmt ein AccessControlList-Objekt entgegen, das eine Liste mit Berechtigten und die festzulegenden Zugriffsebenen enthält.

Importe

```
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;
```

Code

```
    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**
>
> Sie können den eindeutigen Bezeichner des Berechtigten direkt mit der Grantee-Klasse angeben. Verwenden Sie alternativ die EmailAddressGrantee-Klasse, um den Berechtigten per E-Mail festzulegen, wie wir dies hier getan haben.

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- GET Bucket ACLimAmazon S3-API-Referenz
- PUT Bucket ACLimAmazon S3-API-Referenz
- GET Object aclimAmazon S3-API-Referenz
- PUT Object ACLimAmazon S3-API-Referenz

Verwalten des Zugriffs aufAmazon S3Mit Buckets mithilfe von Bucket-Richtlinien

Sie können eine Bucket-Richtlinie zur Verwaltung des Zugriffs auf Ihre Amazon S3-Buckets festlegen, abrufen oder löschen.

Festlegen einer Bucket-Richtlinie

Sie können die Bucket-Richtlinie für einen bestimmten S3-Bucket wie folgt festlegen:

- Aufrufen des AmazonS3-Kunden`setBucketPolicy`und versorgen Sie es mit einemsetBucketPolicyRequest
- Durch direktes Festlegen der Richtlinie unter Verwendung der `setBucketPolicy`-Überladung, die einen Bucket-Namen und einen Richtlinientext (im JSON-Format) entgegen nimmt

Importe

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
```

Code

```
    s3.setBucketPolicy(bucket_name, policy_text);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Verwenden der Policy-Klasse zum Generieren oder Überprüfen einer Richtlinie

Wenn Sie `setBucketPolicy` eine Bucket-Richtlinie übergeben, können Sie die folgenden Aufgaben ausführen:

- Direktes Übergeben der Richtlinie als Zeichenfolge mit Text im JSON-Format
- Erstellen der Richtlinie mit der Policy-Klasse

Bei Verwendung der `Policy`-Klasse müssen Sie sich keine Gedanken über die korrekte Formatierung Ihrer Text-Zeichenfolge machen. Sie können die Richtlinie als JSON-Text von der `Policy`-Klasse erhalten, indem Sie die `toJson`-Methode aufrufen.

Importe

```
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;
```

Code

```
        new Statement(Statement.Effect.Allow)
                .withPrincipals(Principal.AllUsers)
                .withActions(S3Actions.GetObject)
                .withResources(new Resource(
```

```
                        "{region-arn}s3:::" + bucket_name + "/*")));
return bucket_policy.toJson();
```

Die `Policy`-Klasse bietet außerdem eine `fromJson`-Methode, mit der versucht werden kann, eine Richtlinie aus einer übergebenen JSON-Zeichenfolge zu erstellen. Die Methode überprüft die Zeichenfolge und stellt so sicher, dass sich der Text in eine gültige Richtlinienstruktur umwandeln lässt. Sie löst einen Fehler mit einer `IllegalArgumentException` aus, wenn der Richtlinientext ungültig ist.

```
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());
}
```

Mit dieser Technik können Sie eine Richtlinie im Voraus validieren, die Sie aus einer Datei oder anderweitig einlesen.

Auf GitHub finden Sie ein vollständiges Beispiel.

Abrufen einer Bucket-Richtlinie

So rufen Sie die Richtlinie für einenAmazon S3Bucket, rufen Sie den AmazonS3-Kunden an`getBucketPolicy`-Methode übergeben Sie ihr den Namen des Buckets, dessen Richtlinie Sie abrufen möchten.

Importe

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

Code

```
  try {
      BucketPolicy bucket_policy = s3.getBucketPolicy(bucket_name);
```

```
        policy_text = bucket_policy.getPolicyText();
    } catch (AmazonServiceException e) {
        System.err.println(e.getErrorMessage());
        System.exit(1);
    }
```

Wenn der angegebene Bucket nicht vorhanden ist, Sie nicht darauf zugreifen können oder wenn keine Bucket-Richtlinie eingerichtet ist, wird eine `AmazonServiceException` ausgelöst.

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen einer Bucket-Richtlinie

Rufen Sie zum Löschen einer Bucket-Richtlinie die -Methode des AmazonS3--Clients`deleteBucketPolicy`und übergeben Sie ihr den Bucket-Namen.

Importe

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

Code

```
    try {
        s3.deleteBucketPolicy(bucket_name);
    } catch (AmazonServiceException e) {
        System.err.println(e.getErrorMessage());
        System.exit(1);
    }
```

Diese Methode wird auch dann erfolgreich ausgeführt, wenn der Bucket noch nicht über eine Richtlinie verfügt. Wenn Sie den Namen eines Buckets angeben, der noch nicht vorhanden ist oder für den Sie keinen Zugriff haben, wird eine `AmazonServiceException` ausgelöst.

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- Übersicht über die Zugriffsrichtliniensprache imAmazon Simple Storage Service-Benutzerhandbuch

- Beispiele für Bucket-RichtlinienimAmazon Simple Storage Service-Benutzerhandbuch

benutzen TransferManager zumAmazon S3Operationen

Mit der AWS SDK for Java TransferManager -Klasse können Sie zuverlässig Dateien aus der lokalen Umgebung an Amazon S3 übertragen und Objekte von einem S3-Speicherort an einen anderen kopieren. `TransferManager` kann den Fortschritt einer Übertragung abfragen sowie Uploads und Downloads pausieren und fortsetzen.

> **Note**
>
> Bewährte Methode
> Wir empfehlen, dass Sie die Lebenszyklus-Regel AbortIncompleteMultipartUpload für Ihre Amazon S3-Buckets aktivieren.
> Diese Regel weist Amazon S3 an, mehrteilige Uploads abzubrechen, die nicht innerhalb einer bestimmten Anzahl von Tagen nach dem Start abgeschlossen werden. Wenn die festgelegte Höchstdauer überschritten wird, bricht Amazon S3 den Upload ab und löscht dann die unvollständigen Upload-Daten.
> Weitere Informationen finden Sie unterLebenszykluskonfiguration für einen Bucket mit VersioningimAmazon S3-Benutzerhandbuch.

> **Note**
>
> Diese Codebeispiele gehen davon aus, dass Sie das Material inVerwendung derAWS SDK for Javaund haben default konfiguriertAWSAnmeldeinformationen, die die Informationen in verwendenEinrichtenAWSAnmeldeinformationen und Entwicklungsregionaus.

Hochladen von Dateien und Verzeichnissen

TransferManager kann Dateien, Dateilisten und Verzeichnisse auf beliebige hochladenAmazon S3Eimer, die du hastZuvor erstelltaus.

Themen

- Hochladen einer einzelnen Datei
- Hochladen einer Dateiliste

- Upload eines Verzeichnisses

Hochladen einer einzelnen Datei

Rufen Sie TransferManager'suploadMethode zur Verfügung stellenAmazon S3-Bucket-Name, ein Schlüssel-Objekt-) Name und ein Standard-JavaDatei-Objekt, das die hochzuladende Datei darstellt.

Importe

```
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;
```

Code

```
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();
```

Die upload-Methode kehrt sofort zurück und stellt ein Upload-Objekt bereit, mit dem Sie den Übertragungsstatus abrufen oder auf die Fertigstellung warten können.

Siehe .Warten auf die Fertigstellung einer ÜbertragungWeitere Informationen zur Verwendung vonwaitForCompletionum eine Übertragung erfolgreich abzuschließen, bevor Sie TransferManager's aufrufenshutdownNow-Methode. Während Sie darauf warten, dass die

Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter Abrufen des Übertragungsstatus und -fortschritts.

Auf GitHub finden Sie ein vollständiges Beispiel.

Hochladen einer Dateiliste

Rufen Sie den TransferManager auf, um mehrere Dateien auf einmal hochzuladen.`uploadFileList`-Methode mit folgendem:

- Name eines Amazon S3-Buckets
- Schlüsselpräfix, das dem Namen der erstellten Objekte vorangestellt wird (Pfad innerhalb des Buckets, wo die Objekte abgespeichert werden sollen)
- File-Objekt, das das relative Verzeichnis darstellt, von dem aus die Dateipfade erstellt werden sollen
- List-Objekt mit einer Reihe von File-Objekten zum Hochladen

Importe

```
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;
```

Code

```
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();
```

Siehe .Warten auf die Fertigstellung einer ÜbertragungWeitere Informationen zur Verwendung von`waitForCompletion`um eine Übertragung erfolgreich abzuschließen, bevor Sie TransferManager's aufrufen`shutdownNow`-Methode. Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter Abrufen des Übertragungsstatus und -fortschritts.

DieMultipleFileUploadObjekt zurückgegeben von`uploadFileList`Kann zur Abfrage des Übertragungsstatus oder —fortschritts verwendet werden. Weitere Informationen finden Sie unter Abfrage des aktuellen Fortschritts einer Übertragung und Abruf des Übertragungsfortschritts mit einem ProgressListener.

Sie können auch die `MultipleFileUpload`-Methode der `getSubTransfers`-Klasse verwenden, um die einzelnen `Upload`-Objekte für jede zu übertragende Datei zu erhalten. Weitere Informationen finden Sie unter Abruf des Fortschritts von untergeordneten Übertragungen.

Auf GitHub finden Sie ein vollständiges Beispiel.

Upload eines Verzeichnisses

Sie können TransferManager's verwenden`uploadDirectory`-Methode zum Hochladen eines gesamten Dateiverzeichnisses mit der Option, Dateien in Unterverzeichnissen rekursiv zu kopieren. Sie stellen einen Amazon S3-Bucket-Namen, ein S3-Schlüsselpräfix, ein File-Objekt, das das lokale zu kopierende Verzeichnis darstellt, sowie einen `boolean`-Wert bereit, der angibt, ob Sie Unterverzeichnisse rekursiv kopieren möchten (true oder false).

Importe

```
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;
```

Code

```
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();
```

Siehe .Warten auf die Fertigstellung einer ÜbertragungWeitere Informationen zur Verwendung von`waitForCompletion`um eine Übertragung erfolgreich abzuschließen, bevor Sie TransferManager's aufrufen`shutdownNow`-Methode. Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter Abrufen des Übertragungsstatus und -fortschritts.

DieMultipleFileUploadObjekt zurückgegeben von`uploadFileList`Kann zur Abfrage des Übertragungsstatus oder —fortschritts verwendet werden. Weitere Informationen finden Sie unter Abfrage des aktuellen Fortschritts einer Übertragung und Abruf des Übertragungsfortschritts mit einem ProgressListener.

Sie können auch die `MultipleFileUpload`-Methode der `getSubTransfers`-Klasse verwenden, um die einzelnen `Upload`-Objekte für jede zu übertragende Datei zu erhalten. Weitere Informationen finden Sie unter Abruf des Fortschritts von untergeordneten Übertragungen.

Auf GitHub finden Sie ein vollständiges Beispiel.

Herunterladen von Dateien oder Verzeichnissen

Verwenden der TransferManager klasse, um entweder eine einzelne Datei herunterzuladen (Amazon S3object) oder ein Verzeichnis (einAmazon S3Bucket-Name gefolgt von einem Objektpräfix) vonAmazon S3aus.

Themen

- Herunterladen einer einzelnen Datei
- Herunterladen eines Verzeichnisses

Herunterladen einer einzelnen Datei

Verwenden Sie die TransferManager'sdownload-Methode zur Verfügung stelltAmazon S3-Bucket-Name mit dem herunterzuladenden Objekt, den Schlüssel#- (Objekt-) Namen sowie einenDatei-Objekt, das die auf dem lokalen System zu erstellende Datei darstellt.

Importe

```
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;
```

Code

```
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();
```

Siehe .Warten auf die Fertigstellung einer ÜbertragungWeitere Informationen zur Verwendung von`waitForCompletion`um eine Übertragung erfolgreich abzuschließen, bevor Sie TransferManager's aufrufen`shutdownNow`-Methode. Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter Abrufen des Übertragungsstatus und -fortschritts.

Auf GitHub finden Sie ein vollständiges Beispiel.

Herunterladen eines Verzeichnisses

So laden Sie eine Reihe von Dateien mit gemeinsamem key prefix (analog zu einem Verzeichnis eines Dateisystems) von herunterAmazon S3verwenden Sie den TransferManager`downloadDirectory`-Methode. Die Methode nimmt den Amazon S3-Bucket-Namen entgegen, der die herunterzuladenden Objekte enthält, das gemeinsame Objektpräfix aller Objekte sowie ein File-Objekt, das das Verzeichnis darstellt, in das die Dateien auf dem lokalen System heruntergeladen werden sollen. Wenn das angegebene Verzeichnis noch nicht vorhanden ist, wird es erstellt.

Importe

```
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;
```

Code

```
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();
```

Siehe .Warten auf die Fertigstellung einer ÜbertragungWeitere Informationen zur Verwendung von`waitForCompletion`um eine Übertragung erfolgreich abzuschließen, bevor Sie TransferManager's aufrufen`shutdownNow`-Methode. Während Sie darauf warten, dass die Übertragung abgeschlossen ist, können Sie Aktualisierungen zum Status und Fortschritt abfragen oder diese als Ereignisse empfangen. Weitere Informationen finden Sie unter Abrufen des Übertragungsstatus und -fortschritts.

Auf GitHub finden Sie ein vollständiges Beispiel.

Kopieren von Objekten

Sie können ein Objekt von einem S3-Bucket in einen anderen kopieren, indem Sie den TransferManager verwenden Sie den TransferManager`copy`-Methode.

Importe

```
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;
```

Code

```
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();
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Warten auf die Fertigstellung einer Übertragung

Wenn Ihre Anwendung (oder Ihr Thread) blockieren kann, bis die Übertragung abgeschlossen ist, können Sie dieÜbertragung-Schnittstelle`waitForCompletion`-Methode zu blockieren, bis die Übertragung abgeschlossen ist oder eine Ausnahme auftritt.

```
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);
}
```

Sie können den Fortschritt von Übertragungen abrufen, indem Sie Ereignisse abfragen, bevor Sie `waitForCompletion` aufrufen, indem Sie einen Abfragemechanismus in einem separaten Thread implementieren oder Fortschrittsaktualisierungen asynchron mit einem ProgressListener empfangen.

Auf GitHub finden Sie ein vollständiges Beispiel.

Abrufen des Übertragungsstatus und -fortschritt

Jede der vom TransferManager zurückgegebenen Klassen`upload*`,`download*`, und`copy`-Methoden gibt eine Instance der folgenden Klassen zurück, je nachdem, ob es sich um eine Einzel- oder Mehrdateienoperation handelt.

Klasse	Zurückgegeben von
Copy	`copy`
Download	`download`
MultipleFileDownload	`downloadDirectory`
Hochladen	`upload`
MultipleFileUpload	`uploadFileList` , `uploadDirectory`

Alle diese Klassen implementieren die Transfer-Schnittstelle. `Transfer` liefert nützliche Methoden, um den Fortschritt einer Übertragung abzurufen, die Übertragung zu pausieren oder fortzusetzen sowie den aktuellen oder abschließenden Status der Übertragung abzurufen.

Themen

- Abfragen des aktuellen Fortschritts einer Übertragung
- Abruf des Übertragungsfortschritts mit einem ProgressListener
- Abruf des Fortschritts von untergeordneten Übertragungen

Abfragen des aktuellen Fortschritts einer Übertragung

Diese Schleife gibt den Fortschritt einer Übertragung aus, untersucht den aktuellen Fortschritt während der Ausführung und gibt beim Abschluss den abschließenden Status aus.

Importe

```
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;
```

Code

```
// 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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Abruf des Übertragungsfortschritts mit einem ProgressListener

Sie können einenProgressListenerzu jedem Transfer unter Verwendung desÜbertragung-SchnittstelleaddProgressListener-Methode.

Ein ProgressListener erfordert nur eine Methode, nämlich `progressChanged`. Diese nimmt ein ProgressEvent-Objekt entgegen. Mit diesem Objekt können Sie die Gesamtzahl der Bytes der Operation ermitteln, indem Sie die `getBytes`-Methode aufrufen. Die Gesamtzahl der übertragenen Bytes erfahren Sie mit Aufruf von `getBytesTransferred`.

Importe

```
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;
```

Code

```
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();
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Abruf des Fortschritts von untergeordneten Übertragungen

Die MultipleFileUpload-Klasse kann Informationen über untergeordnete Übertragungen zurückgeben, wenn die `getSubTransfers`-Methode aufgerufen wird. Sie gibt zurück, was nicht modifizierbar istSammlungvonHochladen-Objekte, die den jeweiligen Übertragungsstatus und —fortschritt jeder untergeordneten Übertragung angeben.

Importe

```
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;
```

Code

```
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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- ObjektschlüsselimAmazon Simple Storage Service-Benutzerhandbuch

Konfigurieren einesAmazon S3Bucket als Website

Sie können einen Amazon S3-Bucket so konfigurieren, dass er sich wie eine Website verhält. Hierzu müssen Sie die Website-Konfiguration festlegen.

> **Note**
>
> Diese Codebeispiele gehen davon aus, dass Sie das Material in verstehenVerwendung derAWS SDK for Javaund haben default konfiguriertAWSAnmeldeinformationen, die die Informationen in verwendenEinrichtenAWSAnmeldeinformationen und Region für die Entwicklungaus.

Festlegen der Website-Konfiguration eines Buckets

So legen Sie einAmazon S3Bucket Website-Konfiguration, rufen Sie die AmazonS3's an`setWebsiteConfiguration`Methode mit dem Bucket-Namen, für den die Konfiguration festgelegt werden soll, und einemBucketWebSiteConfigurationObjekt, das die Website-Konfiguration des Buckets enthält.

Das Festlegen eines Index-Dokuments ist erforderlich. Alle anderen Parameter sind optional.

Importe

```
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;
```

Code

```
    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**
>
> Beim Festlegen einer Website-Konfiguration werden die Zugriffsberechtigungen für den Bucket nicht geändert. Um die enthaltenen Dateien im Internet sichtbar zu machen, müssen Sie zusätzlich eine Bucket-Richtlinie festlegen, durch die der öffentliche Lesezugriff für die Dateien in dem Bucket ermöglicht wird. Weitere Informationen finden Sie unterVerwalten des Zugriffs aufAmazon S3Buckets Verwenden von -Bucket-Richtlinienaus.

Auf GitHub finden Sie ein vollständiges Beispiel.

Abruf der Website-Konfiguration eines Buckets

Um einen zu bekommenAmazon S3Bucket Website-Konfiguration, rufen Sie die AmazonS3's an`getWebsiteConfiguration`-Methode mit dem Namen des Buckets, für den Sie die Konfiguration abrufen möchten.

Die Konfiguration wird als BucketWebsiteConfiguration-Objekt zurückgegeben. Wenn keine Website-Konfiguration für den Bucket vorhanden ist, wird `null` zurückgegeben.

Importe

```
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;
```

Code

```
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);
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen der Website-Konfiguration eines Buckets

So löschen Sie einenAmazon S3Bucket Website-Konfiguration, rufen Sie die AmazonS3's and`deleteWebsiteConfiguration`-Methode mit dem Namen des Buckets, aus dem Sie die Konfiguration löschen möchten.

Importe

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

Code

```
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);
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- PUT Bucket-WebsiteimAmazon S3-API-Referenz
- GET Bucket-WebsiteimAmazon S3-API-Referenz
- DELETE Bucket-WebsiteimAmazon S3-API-Referenz

Verwenden vonAmazon S3Clientseitige -Verschlüsselung

Das Verschlüsseln von Daten mit dem Amazon S3-Verschlüsselungsclient ist eine Möglichkeit, eine weitere Schutzebene für sensible in Amazon S3 gespeicherte Daten bereitzustellen. Die Beispiele in diesem Abschnitt zeigen, wie Sie einen Amazon S3-Verschlüsselungsclient für Ihre Anwendung erstellen und konfigurieren.

Wenn Sie mit der Kryptografie noch nicht beschäftigt sind, sehen Sie sich dieKryptografiegrundlagenimAWSKMS Developer Guide für eine grundlegende Übersicht über Kryptografie-Begriffe und -algorithmen. Informationen zur Kryptografieunterstützung für alleAWSSDKs finden Sie unterAWSSDK-Support fürAmazon S3Clientseitige VerschlüsselungimAmazon Web ServicesAllgemeine Referenz.

> **Note**
>
> Diese Codebeispiele gehen davon aus, dass Sie das Material inVerwendung derAWS SDK for Javaund haben default konfiguriertAWSAnmeldeinformationen, die die Informationen in verwendenEinrichtenAWSAnmeldeinformationen und Region für die Entwicklungaus.

Wenn Sie Version 1.11.836 oder früher vonAWS SDK for Java, finden Sie unterAmazon S3Clientmigration der Verschlüsselungfür Informationen zur Migration Ihrer Anwendungen auf spätere Versionen. Wenn Sie nicht migrieren können, lesen Siedieses vollständige Beispielauf GitHub:

Andernfalls, wenn Sie die -Version 1.11.837 oder höher verwendenAWS SDK for Java, lesen Sie die unten aufgeführten BeispielthemenAmazon S3Clientseitige Verschlüsselung.

Themen

- Amazon S3Clientseitige -Verschlüsselung mit Clientschlüsseln
- Amazon S3Clientseitige -Verschlüsselung mitAWSKMS verwaltete Schlüssel

Amazon S3Clientseitige -Verschlüsselung mit Clientschlüsseln

In den folgenden Beispielen wirdAmazons3EncryptionClientV2Builder-Klasse zum Erstellen einerAmazon S3Client mit aktivierter clientseitiger -Verschlüsselung. Nach der Aktivierung werden alle Objekte, die Sie über diesen Client in Amazon S3 hochladen, verschlüsselt. Alle Objekte, die Sie über diesen Client von Amazon S3 erhalten, werden automatisch entschlüsselt.

> **Note**
>
> Die folgenden Beispiele zeigen, wie die clientseitige Amazon S3-Verschlüsselung mit von Kunden verwalteten Client-Master-Schlüsseln verwendet wird. So lernen Sie, wie Sie Verschlüsselung verwendenAWSKMS verwaltete Schlüssel finden Sie unterAmazon S3clientseitige -Verschlüsselung mitAWSKMS verwaltete Schlüsselaus.

Bei der Aktivierung der clientseitigen Verschlüsselung stehen zwei Verschlüsselungsmodi zur AuswahlAmazon S3verschlüsselung: streng authentifiziert oder authentifiziert. In den folgenden Abschnitten sehen Sie, wie die unterschiedlichen Modi aktiviert werden. Informationen über die Algorithmen, die bei den verschiedenen Modi eingesetzt werden, finden Sie in der CryptoMode-Definition.

Erforderliche Importe

Importieren Sie für diese Beispiele die folgenden Klassen.

Importe

```
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;
```

Strikte authentifizierte Verschlüsselung

Strikte authentifizierte Verschlüsselung ist der Standardmodus, wenn nein`CryptoMode`ist angegeben.

Um diesen Modus explizit zu aktivieren, geben Sie die`StrictAuthenticatedEncryption`-Wert im`withCryptoConfiguration`-Methode.

> **Note**
>
> Bei Verwendung der clientseitigen authentifizierten Verschlüsselung müssen Sie die neueste Bouncy Castle jar-Datei im Klassenpfad Ihrer Anwendung einschließen.

Code

```
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");
```

Authentifizierte Verschlüsselungsmodus

Beim Modus `AuthenticatedEncryption` wird während der Verschlüsselung ein verbesserter Schlüsselverpackungsalgorithmus angewendet. Bei einer Entschlüsselung in diesem Modus verifiziert der Algorithmus die Integrität des entschlüsselten Objekts und löst eine Ausnahme aus, wenn das

Objekt nicht verifiziert werden kann. Weitere Einzelheiten zur Funktionsweise der authentifizierten Verschlüsselung finden Sie unterAmazon S3Clientseitige authentifizierte VerschlüsselungBlogbeitrag.

> **Note**
>
> Bei Verwendung der clientseitigen authentifizierten Verschlüsselung müssen Sie die neueste Bouncy Castle jar-Datei im Klassenpfad Ihrer Anwendung einschließen.

Zur Aktivierung des Modus geben Sie den `AuthenticatedEncryption`-Wert in der `withCryptoConfiguration`-Methode an.

Code

```
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 S3Clientseitige -Verschlüsselung mitAWSKMS verwaltete Schlüssel

In den folgenden Beispielen wird dieAmazons3EncryptionClientV2Builder-Klasse zum Erstellen einerAmazon S3Client mit aktivierter clientseitiger -Verschlüsselung. Sobald die Konfiguration erfolgt ist, werden alle Objekte, die Sie über diesen Client in Amazon S3 hochladen, verschlüsselt. Alle Objekte, die Sie über diesen Client von Amazon S3 erhalten, sind automatisch entschlüsselt.

> **Note**
>
> In den folgenden Beispielen wird veranschaulicht, wie dieAmazon S3Clientseitige - Verschlüsselung mitAWSKMS verwaltete Schlüssel. Weitere Informationen zur Verwendung der Verschlüsselung mit Ihren eigenen Schlüsseln finden Sie unterAmazon S3Clientseitige - Verschlüsselung mit Client-Master-Schlüsselnaus.

Bei der Aktivierung der clientseitigen Aktivierung stehen zwei Verschlüsselungsmodi zur AuswahlAmazon S3verschlüsselung: streng authentifiziert oder authentifiziert. In den folgenden Abschnitten sehen Sie, wie die unterschiedlichen Modi aktiviert werden. Informationen über die Algorithmen, die bei den verschiedenen Modi eingesetzt werden, finden Sie in der CryptoMode-Definition.

Erforderliche Importe

Importieren Sie für diese Beispiele die folgenden Klassen.

Importe

```
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;
```

Strikte authentifizierte Verschlüsselung

Wenn keine streng authentifizierte Verschlüsselung ist der StandardmodusCryptoModeist angegeben.

Um diesen Modus explizit zu aktivieren, geben Sie dieStrictAuthenticatedEncryption-Wert imwithCryptoConfiguration-Methode.

> **Note**
>
> Bei Verwendung der clientseitigen authentifizierten Verschlüsselung müssen Sie die neueste Bouncy Castle jar-Datei im Klassenpfad Ihrer Anwendung einschließen.

Code

```
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));
```

Rufen Sie die `putObject`-Methode auf dem Amazon S3-Verschlüsselungsclient auf, um Objekte hochzuladen.

Code

```
s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
 with a key created in the {console}");
```

Sie können das Objekt mit demselben Client abrufen. Bei diesem Beispiel wird die `getObjectAsString`-Methode zum Abrufen der gespeicherten Zeichenfolge eingesetzt.

Code

```
System.out.println(s3Encryption.getObjectAsString(bucket_name, ENCRYPTED_KEY3));
```

Authentifizierte Verschlüsselungsmodus

Beim Modus `AuthenticatedEncryption` wird während der Verschlüsselung ein verbesserter Schlüsselverpackungsalgorithmus angewendet. Bei einer Entschlüsselung in diesem Modus verifiziert der Algorithmus die Integrität des entschlüsselten Objekts und löst eine Ausnahme aus, wenn das Objekt nicht verifiziert werden kann. Weitere Details zur Funktionsweise der authentifizierten Verschlüsselung finden Sie imAmazon S3Clientseitige authentifizierte VerschlüsselungBlogbeitrag.

> **Note**
>
> Bei Verwendung der clientseitigen authentifizierten Verschlüsselung müssen Sie die neueste Bouncy Castle jar-Datei im Klassenpfad Ihrer Anwendung einschließen.

Zur Aktivierung des Modus geben Sie den `AuthenticatedEncryption`-Wert in der `withCryptoConfiguration`-Methode an.

Code

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

Konfigurieren vonAWS KMSAuftraggeber

DieAmazon S3-Verschlüsselungsclient erstellt eineAWS KMSClient standardmäßig, es sei denn, einer wird explizit angegeben.

So legen Sie die Region für diese automatisch erstellte festAWS KMSKunde, setze die`awsKmsRegion`aus.

Code

```
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();
```

Alternativ können Sie Ihre eigenen verwendenAWS KMSClient, um den Verschlüsselungsclient zu initialisieren.

Code

```
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-Beispiele unter Verwenden der AWS SDK for Java

Dieser Abschnitt bietet Beispiele für die Programmierung von Amazon SQS mithilfe des AWS SDK for Java.

> **Note**
>
> Die Beispiele enthalten nur den Code, der zur Demonstration jeder Technik nötig ist. Der komplette Beispiel-Code steht auf GitHub bereit. Von dort aus können Sie eine einzelne Quelldatei herunterladen oder das Repository klonen, um alle Beispiele lokal zu erstellen und auszuführen.

Themen

Arbeiten mitAmazon SQSNachrichtenwarteschlangen

Eine Nachrichtenwarteschlange ist der logische Container, der zum zuverlässigen Senden von Nachrichten in Amazon SQS genutzt wird. Es gibt zwei Arten von Warteschlangen: Standard und First-in-First-out-Verfahren (FIFO). Weitere Informationen zu Warteschlangen und die Unterschiede zwischen diesen Typen finden Sie in derAmazon SQSEntwicklerhandbuchaus.

In diesem Thema wird beschrieben, wie Sie Amazon SQS-Warteschlangen erstellen, auflisten, löschen und die URL einer Warteschlange mit AWS SDK for Java abrufen können.

Erstellen einer Warteschlange

Verwenden Sie die von AmazonSQS--Clients`createQueue`Methode zur Verfügung stellt eineCreateQueueRequest-Objekt, das die Parameter der Warteschlange beschreibt.

Importe

```
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;
```

Code

```
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;
    }
}
```

Sie können die vereinfachte Form von `createQueue` verwenden, die nur einen Namen für die Warteschlange benötigt, um eine Standard-Warteschlange zu erstellen.

```
sqs.createQueue("MyQueue" + new Date().getTime());
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Auflisten von Warteschlangen

Listet dieAmazon SQS-Warteschlangen für Ihr Konto, rufen Sie den AmazonSQS-Kunden an`listQueues`-Methode.

Importe

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

Code

```
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);
}
```

Wenn Sie die `listQueues`-Überladung ohne Parameter aufrufen, werden alle Warteschlangen zurückgegeben. Sie können die zurückgegebenen Ergebnisse filtern, indem Sie ein `ListQueuesRequest`-Objekt übergeben.

Importe

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

Code

```
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);
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Abrufen der URL für eine Warteschlange

Rufen Sie den AmazonSQS--Clients an`getQueueUrl`-Methode.

Importe

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

Code

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

```
String queue_url = sqs.getQueueUrl(QUEUE_NAME).getQueueUrl();
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen einer Warteschlange

Stellen Sie die der Warteschlangen bereitURLzu den AmazonSQS-Kunden`deleteQueue`-Methode.

Importe

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

Code

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

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- WieAmazon SQSArbeiten von -WarteschlangenimAmazon SQSEntwicklerhandbuch
- CreateQueueimAmazon SQS-API-Referenz
- GetQueueUrlimAmazon SQS-API-Referenz
- ListQueuesimAmazon SQS-API-Referenz
- DeleteQueuesimAmazon SQS-API-Referenz

Senden, Empfangen und LöschenAmazon SQSNachrichten

In diesem Thema wird beschrieben, wie Sie Amazon SQS-Nachrichten senden, empfangen und löschen. Nachrichten werden immer mit einer SQS-Warteschlange geliefert.

Senden einer Nachricht

Fügen Sie eine einzelne Nachricht zu einerAmazon SQSWarteschlange durch Aufrufen des AmazonSQS-Clients`sendMessage`-Methode. Geben Sie ein SendMessageRequest-Objekt mit der URL der Warteschlange, dem Nachrichtentext und optional einem Verzögerungswert (in Sekunden) an.

Importe

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

Code

```
SendMessageRequest send_msg_request = new SendMessageRequest()
        .withQueueUrl(queueUrl)
        .withMessageBody("hello world")
        .withDelaySeconds(5);
sqs.sendMessage(send_msg_request);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Senden mehrerer Nachrichten gleichzeitig

Sie können mehrere Nachrichten in einer einzigen Anforderung senden. Um mehrere Nachrichten zu senden, verwenden Sie den AmazonSQS-Client`sendMessageBatch`Methode, die eine`SendMessageBatchRequest`enthält die Warteschlangen-URL und eine Liste von -Nachrichten (jeweils ein`SendMessageBatchRequestEntry`) senden. Sie können auch eine optionale Verzögerung pro Nachricht festlegen.

Importe

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
import com.amazonaws.services.sqs.model.SendMessageBatchRequestEntry;
```

Code

```
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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Empfangen von Nachrichten

Abrufen von aktuell in der Warteschlange enthaltenen Nachrichten durch Aufruf des AmazonSQS-Clients`receiveMessage`Übergeben Sie ihr die Warteschlangen-URL. Nachrichten werden als Liste von Message-Objekten zurückgegeben.

Importe

```
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
```

Code

```
List<Message> messages = sqs.receiveMessage(queueUrl).getMessages();
```

Löschen von Nachrichten nach dem Empfangen

Nach dem Erhalt einer Nachricht und der Verarbeitung der Inhalte können Sie die Nachricht aus der Warteschlange löschen, indem Sie die Empfangsnachricht und die Warteschlangen-URL an den AmazonSQS-Client senden`deleteMessage`-Methode.

Code

```
for (Message m : messages) {
    sqs.deleteMessage(queueUrl, m.getReceiptHandle());
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- WieAmazon SQSArbeiten von -WarteschlangenimAmazon SQSEntwicklerhandbuch
- SendMessageimAmazon SQS-API-Referenz
- SendMessageBatchimAmazon SQS-API-Referenz
- ReceiveMessageimAmazon SQS-API-Referenz
- DeleteMessageimAmazon SQS-API-Referenz

Aktivieren von Langabfragen fürAmazon SQS-Nachrichtenwarteschlangen

Amazon SQSVerwendungszweckeKurzabfragensendet standardmäßig anhand einer gewichteten zufälligen Verteilung Abfragen an eine Stichprobenmenge von Servern, um zu bestimmen ob neue Nachrichten für die Antwort verfügbar sind.

Lange Abfragen helfen, Ihre Kosten für die Verwendung zu senkenAmazon SQSindem Sie die Anzahl der leeren Antworten reduzieren, wenn keine Nachrichten vorhanden sind, die zurückgegeben werden können ReceiveMessage an eine gesendete AnfrageAmazon SQSWarteschlange und eliminiert falsche leere Antworten.

> **Note**
>
> Sie können eine Langabfragen-Häufigkeit von1-20 Sekundenaus.

Aktivieren der Langabfrage beim Erstellen einer Warteschlange

So aktivieren Sie lange Abfragen beim Erstellen einesAmazon SQSWarteschlange, setze die`ReceiveMessageWaitTimeSeconds`Attribut auf derCreateQueueRequestObjekt vor dem Aufruf der AmazonSQS-Klasse`createQueue`-Methode.

Importe

```
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;
```

Code

```
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;
    }
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Aktivieren der Langabfrage für eine vorhandene Warteschlange

Zusätzlich zur Aktivierung der Langabfrage bei der Erstellung einer Warteschlange können Sie sie auch für vorhandene Warteschlange aktivieren`ReceiveMessageWaitTimeSeconds`auf dersetQueueAttributesRequestbevor Sie die AmazonSQS-Klasse aufrufen`setQueueAttributes`-Methode.

Importe

```
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Code

```
SetQueueAttributesRequest set_attrs_request = new SetQueueAttributesRequest()
        .withQueueUrl(queue_url)
        .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");
sqs.setQueueAttributes(set_attrs_request);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Aktivieren von Langabfragen beim Nachrichteneingang

Sie können die Langabfrage beim Erhalt einer Nachricht aktivieren, indem Sie die Wartezeit in Sekunden für dieReceiveMessageRequestdie Sie an die AmazonSQS-Klasse liefern`receiveMessage`-Methode.

> **Note**
>
> Sie sollten sicherstellen, dass dieAWSDas Anforderungstimeout des Kunden ist größer als die maximale Langabfragen-Zeit (20 s), sodass Ihr`receiveMessage`Anfragen haben keine Auszeit, während Sie auf das nächste Umfrageereignis warten!

Importe

```
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
```

Code

```
ReceiveMessageRequest receive_request = new ReceiveMessageRequest()
        .withQueueUrl(queue_url)
        .withWaitTimeSeconds(20);
sqs.receiveMessage(receive_request);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- Amazon SQSLangabfrageimAmazon SQSEntwicklerhandbuch
- CreateQueueimAmazon SQS-API-Referenz
- ReceiveMessageimAmazon SQS-API-Referenz
- SetQueueAttributesimAmazon SQS-API-Referenz

Einrichten der Zeitbeschränkung für die Sichtbarkeit inAmazon SQS

Wenn eine Nachricht in Amazon SQS empfangen wird, verbleibt sie bis zum Löschen in der Warteschlange. So wird ihr Erhalt sichergestellt. Eine empfangene, aber nicht gelöschte Nachricht erscheint erst nach Ablauf einer bestimmten Zeitbeschränkung für die Sichtbarkeit in nachfolgenden Anforderungen. Dadurch wird gewährleistet, dass die Nachricht nicht mehrmals empfangen wird, bevor sie verarbeitet und gelöscht werden kann.

> **Note**
>
> Bei Nutzung von Standard-Warteschlangen kann durch die Zeitbeschränkung für die Sichtbarkeit nicht garantiert werden, dass eine Nachricht mehrmals empfangen wird. Wenn Sie eine Standard-Warteschlange verwenden, achten Sie darauf, dass Ihr Code mit dem Fall umgehen kann, dass dieselbe Nachricht mehrmals eingeht.

Einrichten der Zeitbeschränkung für die Sichtbarkeit einer einzelnen Nachricht

Wenn Sie eine Nachricht erhalten haben, können Sie die Zeitbescheinigung für die Sichtbarkeit durch Übergeben ihrer Empfangsnachricht in einerChangeMessageVisibilityRequestdie Sie an die AmazonSQS-Klasse weitergebenchangeMessageVisibility-Methode.

Importe

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

Code

```
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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Einrichten der Zeitbeschränkung für die Sichtbarkeit mehrerer Nachrichten auf einmal

Sie können die Zeitbeschränkung für die Sichtbarkeit mehrerer Nachrichten auf einmal einrichten, indem Sie eine Liste mit ChangeMessageVisibilityBatchRequestEntry-Objekten erstellen. Dabei sollte jedes Objekt eine Zeichenfolge mit eindeutiger ID sowie eine Empfangsnachricht enthalten. Anschließend übergeben Sie die Liste der changeMessageVisibilityBatch-Methode der Amazon SQS-Client-Klasse.

Importe

```
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;
```

Code

```
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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- Zeitbeschränkung für die SichtbarkeitimAmazon SQSEntwicklerhandbuch
- SetQueueAttributesimAmazon SQS-API-Referenz
- GetQueueAttributesimAmazon SQS-API-Referenz
- ReceiveMessageimAmazon SQS-API-Referenz
- ChangeMessageVisibilityimAmazon SQS-API-Referenz
- ChangeMessageVisibilityBatchimAmazon SQS-API-Referenz

Verwenden von Warteschlangen für unzustellbare Nachrichten in Amazon SQS

Amazon SQS bietet Unterstützung für Warteschlangen für unzustellbare Nachrichten. Andere (Quell-)Warteschlangen können Nachrichten, die nicht erfolgreich verarbeitet werden konnten, an die Warteschlange für unzustellbare Nachrichten senden. Sie können diese Nachrichten in der Warteschlange für unzustellbare Nachrichten sammeln und isolieren, um zu bestimmen, warum die Verarbeitung fehlgeschlagen ist.

Erstellen einer Warteschlange für unzustellbare Nachrichten

Eine Warteschlange für unzustellbare Nachrichten wird wie eine reguläre Warteschlange erstellt, hat aber folgende Einschränkungen:

- Eine Warteschlange für unzustellbare Nachrichten muss den gleichen Typ der Warteschlange (FIFO oder Standard) wie die Quell-Warteschlange haben.
- Eine Warteschlange für unzustellbare Nachrichten muss mit demselben erstellt werdenAWS-Kontound region als Quellwarteschlange

Hier erstellen wir zwei identische Amazon SQS-Warteschlangen, von denen eine als Warteschlange für unzustellbare Nachrichten dient:

Importe

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

Code

```
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;
    }
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Zuweisen einer Warteschlange für unzustellbare Nachrichten an eine Quell-Warteschlange

Sie können eine Warteschlange für unzustellbare Nachrichten zuweisen, indem Sie als Erstes eine Redrive-Richtlinie erstellen und die Richtlinie dann in den Attributen der Warteschlange festlegen. Eine Redrive-Richtlinie wird in JSON angegeben und enthält den ARN der Warteschlange für unzustellbare Nachrichten sowie die maximale Anzahl an Malen, die eine Nachricht empfangen und nicht verarbeitet werden kann, bevor sie an die Warteschlange für unzustellbare Nachrichten gesendet wird.

Um die Redrive-Richtlinie für eine Quellwarteschlange festzulegen, rufen Sie die AmazonSQS-Klasse auf`setQueueAttributes`-Methode mit einemsetQueueAttributesRequestObjekt, für das Sie den`RedrivePolicy`-Attribut mit Ihrer JSON-Redrive-Richtlinie.

Importe

```
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Code

```
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);
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- benutzenAmazon SQSDead-Letter-QueuesimAmazon SQSEntwicklerhandbuch
- SetQueueAttributesimAmazon SQS-API-Referenz

Amazon SWF-Beispiele unter Verwenden der AWS SDK for Java

Amazon SWFist ein Dienst zur Verwaltung von Workflows und hilft Entwicklern bei der Erstellung und Skalierung verteilter Workflows. Diese können parallele oder sequenzielle Schritte mit Aktivitäten, untergeordneten Workflows und sogarLambda-Aufgaben.

Es gibt zwei Möglichkeiten, mit zu arbeitenAmazon SWFVerwendung vonAWS SDK for Java, unter Verwendung des SWFAuftraggeberObjekt oder durch Verwendung desAWS Flow Frameworkfür Java. DieAWS Flow Frameworkfür Java ist anfangs schwieriger zu konfigurieren, da dabei in hohem Maß Anmerkungen verwendet werden. Außerdem werden zusätzliche Bibliotheken verwendet, wie z. B. AspectJ und das Spring Framework. Bei großen oder komplexen Projekten sparen Sie allerdings Entwicklungszeit, wenn Sie verwendenAWS Flow Frameworkfür Java. Weitere Informationen finden Sie im .AWS Flow Frameworkfür Java-Entwicklerhandbuchaus.

Dieser Abschnitt zeigt anhand von Beispielen, wie sich Amazon SWF durch direkte Nutzung des AWS SDK for Java-Clients programmieren lässt.

Themen

Grundlagen der SWF

Hier werden häufige Aufgaben bei der Arbeit mit Amazon SWF mithilfe des AWS SDK for Java beschrieben. Dies soll hauptsächlich als Referenz dienen. Eine vollständigere Anleitung für die Einleitung finden Sie unterErstellen einer einfachenAmazon SWFAnwendungaus.

Abhängigkeiten

Einfache Amazon SWF-Anwendungen erfordern die folgenden Abhängigkeiten, die im AWS SDK for Java enthalten sind:

- aws-java-sdk-1.12.*.jar
- commons-logging-1.2.*.jar
- httpclient-4.3.*.jar
- httpcore-4.3.*.jar
- jackson-annotations-2.12.*.jar
- jackson-core-2.12.*.jar
- jackson-databind-2.12.*.jar
- joda-time-2.8.*.jar

> **Note**
>
> Die Versionsnummern dieser Pakete unterscheiden sich je nach verwendeter SDK-Version. Die mit dem SDK mitgelieferten Versionen wurden allerdings auf Kompatibilität getestet, daher sollten Sie diese Versionen nutzen.

AWS Flow FrameworkFür Java-Anwendungen erfordern zusätzliche Einrichtung,undzusätzliche Abhängigkeiten. Sehen Sie dieAWS Flow Frameworkfor Java-EntwicklerhandbuchWeitere Informationen zur Verwendung des Frameworks finden Sie unter.

Importe

Im Allgemeinen können Sie die folgenden Importe für Code-Entwicklung nutzen:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Sie sollten jedoch nur die Klassen importieren, die Sie wirklich benötigen.
Dazu geben Sie wahrscheinlich bestimmte Klassen im Workspace `com.amazonaws.services.simpleworkflow.model` an:

```
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;
```

Wenn Sie verwendenAWS Flow FrameworkFür Java importieren Sie Klassen aus`com.amazonaws.services.simpleworkflow.flow`workspace. Beispiel:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.flow.ActivityWorker;
```

> **Note**
>
> DieAWS Flow FrameworkFür Java hat zusätzliche Anforderungen außer denen der BasisAWS SDK for Javaaus. Weitere Informationen finden Sie im .AWS Flow Frameworkfor Java-Entwicklerhandbuchaus.

Verwenden der SWF-Client-Klasse

Prinzipiell greifen Sie auf Amazon SWF entweder über die Klasse AmazonSimpleWorkflowClient oder über die Klasse AmazonSimpleWorkflowAsyncClient zu. Der Unterschied zwischen diesen Klassen besteht darin, dass die `\*AsyncClient`-Klasse Future-Objekte für gleichzeitige (asynchrone) Programmierung zurückgibt.

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

Eine einfache Amazon SWF Anwendung erstellen

In diesem Thema werden Sie in die Programmierung von Amazon SWFAnwendungen mit dem AWS SDK for Java eingeführt und nebenbei einige wichtige Konzepte vorgestellt.

Über das Beispiel

Im Beispielprojekt wird ein Workflow mit einer einzigen Aktivität erstellt, die Workflow-Daten akzeptiert, die über die AWS Cloud übertragen werden (in der Tradition von ist dies der Name einer PersonHelloWorld, die begrüßt werden soll), und druckt dann als Antwort eine Begrüßung aus.

Oberflächlich betrachtet scheint dies sehr einfach zu sein. Im Hintergrund bestehen Amazon SWF-Anwendungen aus mehreren Teilen, die zusammenarbeiten:

- Einer Domäne als logischem Container für die Ausführungsdaten des Workflows.
- Einem oder mehreren Workflows, die Code-Komponenten darstellen, mit denen die logische Reihenfolge der Ausführung für die Aktivitäten und untergeordneten Workflows Ihres Workflows definiert wird.
- Einem Workflow-Worker, auch Entscheider genannt, der Abfragen für Entscheidungsaufgaben ausführt und daraufhin Aktivitäten oder untergeordnete Workflows plant.
- Einer oder mehreren Aktivitäten, die jeweils eine Arbeitseinheit im Workflow darstellen.
- Einem Aktivitäts-Worker, der Abfragen für Aktivitätsaufgaben durchführt und als Reaktion Aktivitätsmethoden ausführt.
- Einer oder mehreren Aufgabenlisten, also von Amazon SWF verwaltete Warteschlangen zur Ausstellung von Anforderungen an die Workflow- und Aktivitäts-Worker. Aufgaben in einer Aufgabenliste für Workflow-Worker werden Entscheidungsaufgaben genannt. Aufgaben für Aktivitäts-Worker nennen sich Aktivitätsaufgaben.
- Einem Workflow-Starter, der mit der Ausführung des Workflows beginnt.

Im Hintergrund Amazon SWF orchestriert er den Betrieb dieser Komponenten, koordiniert ihren Ablauf aus der AWS Cloud, überträgt Daten zwischen ihnen, verarbeitet Timeouts und Heartbeat-Benachrichtigungen und protokolliert den Verlauf der Workflow-Ausführung.

Voraussetzungen

Entwicklungsumgebung

Die Entwicklungsumgebung in dieser Anleitung besteht aus:

- Das Tool AWS SDK for Java.
- Apache Maven (3.3.1).
- JDK 1.7 oder neuer. Diese Anleitung wurde mit JDK 1.8.0 entwickelt und getestet.
- Einen guten Java-Texteditor (Ihrer Wahl).

> **Note**
>
> Wenn Sie ein anderes Build-System als Maven verwenden, können Sie trotzdem ein Projekt erstellen, indem Sie die entsprechenden Schritte für Ihre Umgebung verwenden und die hier bereitgestellten Konzepte verwenden, um weiterzumachen. Weitere Informationen zur Konfiguration und Verwendung von AWS SDK for Java mit verschiedenen Build-Systemen finden Sie unter Erste Schritte.
> Ebenso, aber mit mehr Aufwand, können die hier gezeigten Schritte mit jedem der AWS SDKs implementiert werden, die Unterstützung für Amazon SWF bieten.

Alle erforderlichen externen Abhängigkeiten sind in AWS SDK for Java enthalten. Sie müssen nichts zusätzlich herunterladen.

AWSZugriff

Um dieses Tutorial erfolgreich durcharbeiten zu können, müssen Sie Zugriff auf das AWS Zugriffsportal haben, wie im Abschnitt „Grundeinstellungen“ dieses Handbuchs beschrieben.

In den Anweisungen wird beschrieben, wie Sie auf temporäre Anmeldeinformationen zugreifen, die Sie kopieren und in Ihre lokale gemeinsame `credentials` Datei einfügen. Die temporären Anmeldeinformationen, die Sie einfügen, müssen mit einer IAM-Rolle verknüpft seinAWS IAM Identity Center, die über Zugriffsberechtigungen für Amazon SWF verfügt. Nach dem Einfügen der temporären Anmeldeinformationen sieht etwa folgendermaßen aus. `credentials`

```
[default]
```

```
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONGSTRI
```

Diese temporären Anmeldeinformationen sind mit dem `default` Profil verknüpft.

Erstellen eines SWF-Projekts

1. Starten Sie ein neues Projekt mit Maven:

   ```
   mvn archetype:generate -DartifactId=helloswf \
   -DgroupId=aws.example.helloswf -DinteractiveMode=false
   ```

   Dadurch wird ein neues Projekt mit einer Standard-Projektstruktur für Maven erzeugt:

   ```
   helloswf
   ### pom.xml
   ### src
       ### main
       #   ### java
       #       ### aws
       #           ### example
       #               ### helloswf
       #                   ### App.java
       ### test
           ### ...
   ```

   Sie können das `test`-Verzeichnis und dessen Inhalte ignorieren oder löschen, da wir es für diese Anleitung nicht verwenden. Sie können auch `App.java` löschen, da wir sie durch neue Klassen ersetzen.

2. Bearbeiten Sie die `pom.xml` Projektdatei und fügen Sie das aws-java-sdk-simpleworkflowModul hinzu, indem Sie innerhalb des `<dependencies>` Blocks eine Abhängigkeit dafür hinzufügen.

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

3. Achten Sie darauf, dass Maven das Projekt mit Unterstützung für JDK 1.7+ erstellt. Fügen Sie Folgendes in der Datei `<dependencies>` zu Ihrem Projekt hinzu (vor oder nach dem Block `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>
```

Entwickeln des Projekts

Das Beispielprojekt umfasst vier separate Anwendungen, die wir einzeln untersuchen:

- HelloTypes.java — enthält die Domänen-, Aktivitäts- und Workflow-Typdaten des Projekts, die mit den anderen Komponenten geteilt werden. Außerdem übernimmt diese Datei das Registrieren dieser Typen mit SWF.
- ActivityWorker.java — enthält den Activity Worker, der nach Aktivitätsaufgaben abfragt und daraufhin Aktivitäten ausführt.
- WorkflowWorker.java — enthält den Workflow-Worker (Entscheider), der nach Entscheidungsaufgaben fragt und neue Aktivitäten plant.
- WorkflowStarter.java — enthält den Workflow-Starter, der eine neue Workflow-Ausführung startet, wodurch SWF mit der Generierung von Entscheidungs- und Workflow-Aufgaben beginnt, die Ihre Mitarbeiter bearbeiten können.

Allgemeine Schritte für alle Quelldateien

Alle Dateien, die Sie erstellen, um Ihre Java-Klassen zu integrieren, haben ein paar Dinge gemeinsam. Aus zeitlichen Gründen werden die folgenden Schritte jedes Mal implizit vorausgesetzt, wenn Sie eine neue Datei zum Projekt hinzufügen:

1. Erstellen Sie die Datei im Verzeichnis `src/main/java/aws/example/helloswf/` des Projekts.
2. Fügen Sie eine `package`-Deklaration am Anfang jeder Datei hinzu, um ihren Namespace zu deklarieren. Das Beispielprojekt nutzt:

```
package aws.example.helloswf;
```

3. Fügen Sie `import` Deklarationen für die AmazonSimpleWorkflowClientKlasse und für mehrere Klassen im `com.amazonaws.services.simpleworkflow.model` Namespace hinzu. Der Einfachheit halber verwenden wir:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Registrieren von Domäne und Workflow- und Aktivitätstypen

Als Erstes erstellen wir eine neue ausführbare Klasse, `HelloTypes.java`. Diese Datei enthält freigegebene Daten, die verschiedenen Teilen Ihres Workflows bekannt sein müssen, z. B. die Namen und Version Ihrer Aktivitäten und Workflow-Typen, den Namen der Domäne und den Namen der Aufgabenliste.

1. Öffnen Sie den Texteditor und erstellen Sie die Datei `HelloTypes.java`. Fügen Sie eine Package-Deklaration und die Importe laut den allgemeinen Schritten hinzu.
2. Deklarieren Sie die `HelloTypes`-Klasse und geben Sie Werte für Ihre registrierten Aktivitäts- und Workflow-Typen an:

```
    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";
```

   Diese Werte werden im gesamten Code verwendet.

3. Erstellen Sie nach den String-Deklarationen eine Instanz der AmazonSimpleWorkflowClientKlasse. Dies ist die grundlegende vom Amazon SWF bereitgestellte Schnittstelle zu den AWS SDK for Java-Methoden.

```
private static final AmazonSimpleWorkflow swf =

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

   Das vorherige Snippet geht davon aus, dass temporäre Anmeldeinformationen mit dem `default` Profil verknüpft sind. Wenn Sie ein anderes Profil verwenden, ändern Sie den obigen Code wie folgt und ersetzen Sie *`profile_name`* durch den Namen des tatsächlichen Profilnamens.

```
private static final AmazonSimpleWorkflow swf =
        AmazonSimpleWorkflowClientBuilder
                .standard()
                .withCredentials(new ProfileCredentialsProvider("profile_name"))
                .withRegion(Regions.DEFAULT_REGION)
                .build();
```

4. Fügen Sie eine neue Funktion für die Registrierung einer SWF-Domäne hinzu. Bei einer Domäne handelt es sich um einen logischen Container für eine Reihe von zugehörigen SWF-Aktivitäten und Workflow-Typen. SWF-Komponenten können nur miteinander kommunizieren, wenn sie in derselben Domäne sind.

```
    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!");
    }
```

   Wenn Sie eine Domain registrieren, geben Sie ihr einen Namen (beliebiger Satz von 1 bis 256 Zeichen mit Ausnahme von`:`,, `/|`, Steuerzeichen oder der literalen Zeichenfolge '`arn') und einen Aufbewahrungszeitraum an. Dies ist die Anzahl der Tage, an denen die Ausführungsdaten Ihres Workflows nach Abschluss einer Workflow-Ausführung gespeichert Amazon SWF werden. Der maximale Aufbewahrungszeitraum für die Workflow-Ausführung ist 90 Tage. RegisterDomainRequestWeitere Informationen finden Sie unter.

Wenn eine Domain mit diesem Namen bereits existiert, DomainAlreadyExistsExceptionwird a ausgelöst. Da uns nicht interessiert, ob die Domäne schon erstellt wurde, können wir die Ausnahme ignorieren.

> **Note**
>
> Dieser Code zeigt ein gemeinsames Muster beim Umgang mit AWS SDK for Java-Methoden: Daten für die Methode werden von einer Klasse im `simpleworkflow.model`-Namespace bereitgestellt, die Sie mit den verkettbaren `0with*`-Methoden instanziieren und befüllen.

5. Fügen Sie eine neue Funktion für die Registrierung eines neuen Aktivitätstyps hinzu. Eine Aktivität stellt eine Arbeitseinheit in Ihrem Workflow dar.

```
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!");
}
```

Ein Aktivitätstyp wird durch einen Namen und eine Version angegeben, die zum Unterscheiden der Aktivitäten von denen anderen Dateien in der Domäne, in der sie registriert sind, verwendet werden. Aktivitäten enthalten außerdem eine Reihe von optionalen Parametern, wie die Standard-Aufgabenliste für den Empfang von Aufgaben und Daten aus SWF und eine Reihe verschiedener Timeouts, mit denen Sie Einschränkungen dafür, wie lange verschiedene Teile der Aktivität ausgeführt werden dürfen, festlegen können. RegisterActivityTypeRequestWeitere Informationen finden Sie unter.

> **Note**
>
> Alle Timeout-Werte werden in Sekunden angegeben. Eine vollständige Beschreibung der Auswirkungen von Amazon SWFTimeouts auf Ihre Workflow-Ausführungen finden Sie unter Timeout-Typen.

Wenn der Aktivitätstyp, den Sie registrieren möchten, bereits existiert, TypeAlreadyExistsExceptionwird eine ausgelöst. Fügen Sie eine neue Funktion für die Registrierung eines neuen Workflow-Typs hinzu. Ein Workflow, auch Entscheider genannt, stellt die Logik Ihrer Workflow-Ausführung dar.

+

```
    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!");
    }
```

+

Ähnlich wie bei Aktivitätstypen werden Workflow-Typen durch einen Namen und eine Version identifiziert und haben auch konfigurierbare Timeouts. RegisterWorkflowTypeRequestWeitere Informationen finden Sie unter.

+

Wenn der Workflowtyp, den Sie registrieren möchten, bereits existiert, TypeAlreadyExistsExceptionwird ein ausgelöst. Markieren Sie die Klasse schließlich als ausführbar, indem Sie eine `main`-Methode hinzufügen. Diese registriert die Domäne, den Aktivitätstyp sowie den Workflow-Typ:

+

```
    registerDomain();
    registerWorkflowType();
    registerActivityType();
```

Jetzt können Sie die Anwendung erstellen und ausführen, um das Registrierungsskript auszuführen. Sie können aber auch mit dem Entwickeln der Aktivitäts- und Workflow-Worker fortfahren. Sobald die Domain, der Workflow und die Aktivität registriert wurden, müssen Sie dies nicht erneut ausführen. Diese Typen bleiben bestehen, bis Sie sie selbst als veraltet markieren.

Implementieren des Aktivitäts-Workers

Eine Aktivität ist die grundlegende Arbeitseinheit in einem Workflow. Ein Workflow stellt die Logik bereit und plant auszuführende Aktivitäten (oder andere Aktionen) als Reaktion auf Entscheidungsaufgaben. Ein typischer Workflow besteht normalerweise aus einer Reihe von Aktivitäten, die synchron, asynchron oder gemischt ausgeführt werden können.

Der Aktivitäts-Worker ist der Codeteil, der Abfragen für Aktivitätsaufgaben durchführt. Diese werden von Amazon SWF als Reaktion auf Workflow-Entscheidungen generiert. Wird eine Aktivitätsaufgabe empfangen, wird die zugehörige Aktivität ausgeführt und eine Erfolg-/Fehlermeldung an den Workflow zurückgegeben.

Wir implementieren einen einfachen Aktivitäts-Worker, der eine einzelne Aktivität ausführt.

1. Öffnen Sie den Texteditor und erstellen Sie die Datei `ActivityWorker.java`. Fügen Sie eine Package-Deklaration und die Importe laut den allgemeinen Schritten hinzu.

   ```
   import com.amazonaws.regions.Regions;
   import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
   import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
   import com.amazonaws.services.simpleworkflow.model.*;
   ```

2. Fügen Sie der Datei die `ActivityWorker`-Klasse hinzu und ergänzen Sie darin ein Datenmitglied, in dem ein SWF-Client gespeichert werden kann. Darüber interagieren wir mit Amazon SWF:

   ```
       private static final AmazonSimpleWorkflow swf =

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

3. Fügen Sie die Methode, die wir nutzen werden, als Aktivität hinzu:

```
private static String sayHello(String input) throws Throwable {
    return "Hello, " + input + "!";
}
```

   Die Aktivität nimmt einfach eine Zeichenfolge entgegen, kombiniert sie zu einer Begrüßung und gibt das Ergebnis zurück. Es ist zwar unwahrscheinlich, dass diese Aktivität eine Ausnahme auslöst. Dennoch empfiehlt es sich, Aktivitäten zu entwerfen, die einen Fehler auslösen können, wenn ein Fehler auftritt.

4. Fügen Sie eine `main`-Methode hinzu. Wir verwenden sie als Abfragemethode der Aktivitätsaufgabe. Wir starten sie, indem wir Code hinzufügen, der die Aufgabenliste nach Aktivitätsaufgaben abfragt:

```
        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();
```

   Die Aktivität erhält Aufgaben von, Amazon SWF indem sie die `pollForActivityTask` Methode des SWF-Clients aufruft und dabei die Domäne und Aufgabenliste angibt, die in der übergebenen Datei verwendet werden sollen. PollForActivityTaskRequest

   Sobald eine Aufgabe empfangen wird, rufen wir eine eindeutige Kennung für sie ab, indem wir die `getTaskToken`-Methode der Aufgabe aufrufen.

5. Schreiben Sie als Nächstes Code zum Verarbeiten der eingehenden Aufgaben. Fügen Sie Folgendes zur `main`-Methode hinzu, und zwar direkt nach dem Code, der die Aufgabe abruft und deren Aufgabentoken ermittelt.

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

Wenn das Aufgabentoken ungleich `null` ist, beginnen wir die Ausführung der Aktivitätsmethode (`sayHello`) und übergeben dabei die Eingabedaten, die mit der Aufgabe mitgesendet wurden.

Wenn die Aufgabe erfolgreich war (es wurde kein Fehler generiert), reagiert der Worker auf SWF, indem er die `respondActivityTaskCompleted` Methode des SWF-Clients mit einem RespondActivityTaskCompletedRequestObjekt aufruft, das das Task-Token und die Ergebnisdaten der Aktivität enthält.

Wenn die Aufgabe jedoch fehlschlägt, antworten wir, indem wir die `respondActivityTaskFailed` Methode mit einem RespondActivityTaskFailedRequestObjekt aufrufen und ihr das Task-Token und Informationen über den Fehler übergeben.

> **Note**
>
> Diese Aktivität wird nicht korrekt beendet, wenn sie unsanft abgebrochen wird. Dies geht zwar über die Grenzen dieser Anleitung hinaus, doch eine alternative Implementierung dieses Aktivitäts-Workers finden Sie im begleitenden Thema Korrektes Herunterfahren von Aktivitäts- und Workflow-Workern.

Implementieren des Workflow-Workers

Die Workflow-Logik liegt in einem Codeteil, der Workflow-Worker genannt wird. Der Workflow-Worker ruft die von Amazon SWF in der Domäne und auf der Standard-Aufgabenliste, für die der Workflow-Typ registriert wurde, gesendeten Entscheidungsaufgaben ab.

Wenn der Workflow-Worker eine Aufgabe erhält, wird eine Art Entscheidung gefällt (in der Regel, ob eine neue Aktivität geplant werden soll oder nicht) und eine entsprechende Aktion ausgeführt (z. B. zur Planung der Aktivität).

1. Öffnen Sie den Texteditor und erstellen Sie die Datei `WorkflowWorker.java`. Fügen Sie eine Package-Deklaration und die Importe laut den allgemeinen Schritten hinzu.
2. Fügen Sie einige zusätzliche Importe in die Datei ein:

```
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. Deklarieren Sie die `WorkflowWorker` Klasse und erstellen Sie eine Instanz der AmazonSimpleWorkflowClientKlasse, die für den Zugriff auf SWF-Methoden verwendet wird.

```
    private static final AmazonSimpleWorkflow swf =

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

4. Fügen Sie die `main`-Methode hinzu. Die Methode wird in Schleife ausgeführt und ruft Entscheidungsaufgaben mit der `pollForDecisionTask`-Methode des SWF-Clients ab. Der PollForDecisionTaskRequestliefert die Details.

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

Sobald eine Aufgabe empfangen wird, rufen wir ihre `getTaskToken`-Methode auf. Diese gibt eine Zeichenfolge zur Erkennung der Aufgabe zurück. Wenn das zurückgegebene Token nicht `null` vorhanden ist, verarbeiten wir es in der `executeDecisionTask` Methode weiter und übergeben ihm das Task-Token und die Liste der mit der Aufgabe gesendeten HistoryEventObjekte.

5. Fügen Sie die `executeDecisionTask`-Methode hinzu. Sie nimmt das Aufgabentoken (einen `String`) und eine `HistoryEvent`-Liste entgegen.

```
    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;
```

Wir richten auch einige Datenmitglieder zur Nachverfolgung ein, u. a.:

- Eine Liste mit Decision-Objekten, mit denen die Ergebnisse der Aufgabenverarbeitung berichtet werden.
- Eine Zeichenfolge zur Speicherung der Workflow-Eingabe, die durch das Ereignis "WorkflowExecutionStarted" bereitgestellt wird
- Eine Zählung der geplanten und offenen (aktiven) Aktivitäten. So wird die Planung von Aktivitäten vermieden, die bereits geplant wurden oder momentan ausgeführt werden.
- Einen boolescher Wert, der angibt, ob die Aktivität abgeschlossen ist.
- Eine Zeichenfolge, die die Aktivitätsergebnisse für die Rückgabe als unser Workflow-Ergebnis speichert.

6. Fügen Sie als Nächstes Code in die `executeDecisionTask`-Methode ein, der die mit der Aufgabe mitgesendeten `HistoryEvent`-Objekte verarbeitet, je nachdem, welcher Ereignistyp von der `getEventType`-Methode gemeldet wurde.

```
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("]");
```

Für die Zwecke unseres Workflows interessieren wir uns am meisten für:

- das Ereignis WorkflowExecutionStarted "", das angibt, dass die Workflow-Ausführung gestartet wurde (was in der Regel bedeutet, dass Sie die erste Aktivität im Workflow ausführen sollten) und das die erste Eingabe für den Workflow bereitstellt. In diesem Fall handelt es sich um den Namen für unsere Begrüßung. Deswegen speichern wir die Daten in einer Zeichenfolge, während wir die auszuführende Aktivität planen.
- das Ereignis ActivityTaskCompleted "", das gesendet wird, sobald die geplante Aktivität abgeschlossen ist. Die Ereignisdaten enthalten auch den Rückgabewert der abgeschlossenen Aktivität. Da wir nur eine Aktivität haben, verwenden wir diesen Wert als Ergebnis des gesamten Workflows.

Die anderen Ereignistypen können verwendet werden, wenn Ihre Workflows es erfordern. Informationen zu den einzelnen Ereignistypen finden Sie in der HistoryEventKlassenbeschreibung.

+ HINWEIS: Zeichenketten in `switch` Anweisungen wurden in Java 7 eingeführt. Wenn Sie eine frühere Version von Java verwenden, können Sie die EventTypeKlasse verwenden, um den `String` zurückgegebenen Wert von `history_event.getType()` in einen Enum-Wert und dann bei `String` Bedarf wieder in einen zu konvertieren:

```
EventType et = EventType.fromValue(event.getEventType());
```

1. Fügen Sie nach der `switch`-Anweisung weiteren Code hinzu, um mit einer passenden Entscheidung auf die empfangene Aufgabe zu reagieren.

```
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);
```

- Wenn die Aktivität noch nicht geplant wurde, antworten wir mit einer `ScheduleActivityTask` Entscheidung, die in einer ScheduleActivityTaskDecisionAttributesStruktur Informationen über die Aktivität enthält, die als nächstes geplant Amazon SWF werden soll, einschließlich aller Daten, die an die Aktivität gesendet Amazon SWF werden sollen.
- Wenn die Aktivität abgeschlossen wurde, betrachten wir den gesamten Workflow als abgeschlossen und antworten mit einer `CompletedWorkflowExecution` Entscheidung, indem wir eine CompleteWorkflowExecutionDecisionAttributesStruktur ausfüllen, die Details zum abgeschlossenen Workflow enthält. In diesem Fall geben wir das Ergebnis der Aktivität zurück.

In beiden Fällen werden die Entscheidungsinformationen zur `Decision`-Liste hinzugefügt, die oben in der Methode deklariert wurde.

2. Vervollständigen Sie die Entscheidungsaufgabe, indem Sie die Liste der `Decision`-Objekte zurückgeben, die bei der Verarbeitung der Aufgabe erfasst wurden. Fügen Sie den Code am Ende der `executeDecisionTask`-Methode hinzu, die wir schreiben:

```
swf.respondDecisionTaskCompleted(
    new RespondDecisionTaskCompletedRequest()
        .withTaskToken(taskToken)
        .withDecisions(decisions));
```

Die `respondDecisionTaskCompleted`-Methode des SWF-Clients nimmt das Aufgabentoken zur Erkennung der Aufgabe sowie die Liste der `Decision`-Objekte entgegen.

Implementieren des Workflow-Starters

Schließlich erstellen wir Code zum Starten der Workflow-Ausführung.

1. Öffnen Sie den Texteditor und erstellen Sie die Datei `WorkflowStarter.java`. Fügen Sie eine Package-Deklaration und die Importe laut den allgemeinen Schritten hinzu.
2. Fügen Sie die `WorkflowStarter`-Klasse hinzu:

```
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() + "'.");
    }
}
```

Die `WorkflowStarter`-Klasse besteht aus einer einzelnen Methode `main`, die ein optionales Argument entgegen nimmt. Dieses wird auf der Befehlszeile als Eingabedaten für den Workflow übergeben.

Die SWF-Client-Methode,`startWorkflowExecution`, verwendet ein StartWorkflowExecutionRequestObjekt als Eingabe. Zusätzlich zur Angabe der Domäne und des auszuführenden Workflow-Typs geben wir hier Folgendes an:

- einen lesbaren Namen für die Workflow-Ausführung,
- Workflow-Eingabedaten (in unserem Beispiel auf der Befehlszeile angegeben) sowie
- einen Timeout-Wert, der in Sekunden angibt, wie lange die Ausführung des gesamten Workflows dauern darf.

Das Run-Objekt, das von `startWorkflowExecution` zurückgegeben wird, stellt eine Ausführungs-ID bereit. Mit diesem Wert lässt sich diese bestimmte Workflow-Ausführung im Amazon SWF-Verlauf der Workflow-Ausführungen identifizieren.

+ HINWEIS: Die Run-ID wird von Amazon SWF generiert und entspricht nicht dem Namen der Workflow-Ausführung, den Sie beim Starten der Workflow-Ausführung eingeben.

Erstellen des Beispiels

Sie können das Beispielprojekt mit Maven erstellen, indem Sie zum `helloswf`-Verzeichnis wechseln und Folgendes eingeben:

```
mvn package
```

Die resultierende `helloswf-1.0.jar`-Datei wird im `target`-Verzeichnis erstellt.

Ausführen des Beispiels

Das Beispiel besteht aus vier separaten ausführbaren Klassen, die unabhängig voneinander ausgeführt werden.

> **Note**
>
> Wenn Sie ein Linux-, macOS- oder Unix-System verwenden, können Sie alle nacheinander in einem einzigen Terminalfenster ausführen. Wenn Sie Windows verwenden, sollten Sie zwei weitere Instances der Eingabeaufforderung öffnen und in jedem Fenster zum `helloswf`-Verzeichnis wechseln.

Festlegen des Java-Klassenpfads

Obwohl Maven die Abhängigkeiten für Sie verwaltet hat, müssen Sie zur Ausführung des AWS Beispiels die SDK-Bibliothek und ihre Abhängigkeiten in Ihrem Java-Klassenpfad bereitstellen. Sie können die `CLASSPATH` Umgebungsvariable entweder auf den Speicherort Ihrer AWS SDK-Bibliotheken und das `third-party/lib` Verzeichnis im SDK setzen, das die erforderlichen Abhängigkeiten enthält:

```
export CLASSPATH='target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/
lib/*'
java example.swf.hello.HelloTypes
```

oder verwenden Sie die `-cp` Option des **`java`** Befehls, um den Klassenpfad festzulegen, während die einzelnen Anwendungen ausgeführt werden.

```
java -cp target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/* \
  example.swf.hello.HelloTypes
```

Welche Version Sie bevorzugen, liegt an Ihnen. Wenn Sie keine Probleme beim Erstellen des Codes hatten, versuchen Sie beide, die Beispiele auszuführen, und Sie erhalten eine Reihe von "NoClassDefFound" Fehlern. Dies liegt wahrscheinlich daran, dass der Klassenpfad falsch gesetzt ist.

Registrieren von Domäne und Workflow- und Aktivitätstypen

Vor der Ausführung Ihrer Worker und des Workflow-Starters müssen Sie die Domäne und die Workflow- und Aktivitätstypen registrieren. Der entsprechende Code wurde in den Arbeitsabläufen und Aktivitätstypen registrieren einer Domain implementiert.

Nach dem Erstellen und Festlegen des CLASSPATH können Sie den Code zur Registrierung mit folgendem Befehl ausführen:

```
    echo 'Supply the name of one of the example classes as an argument.'
```

Starten der Aktivitäts- und Workflow-Worker

Nachdem die Typen nun registriert sind, können Sie die Aktivitäts- und Workflow-Worker starten. Diese werden weiter ausgeführt und nach Aufgaben abfragen, bis sie beendet werden. Sie sollten sie also entweder in separaten Terminalfenstern ausführen, oder, wenn Sie unter Linux, macOS oder Unix laufen, können Sie den & Operator verwenden, um jeden von ihnen dazu zu bringen, bei der Ausführung einen separaten Prozess zu starten.

```
    echo 'If there are arguments to the class, put them in quotes after the class
name.'
    exit 1
```

Wenn Sie diese Befehle in separaten Fenstern laufen lassen, lassen Sie den letzten &-Operator in jeder Zeile weg.

Starten der Workflow-Ausführung

Nachdem die Aktivitäts- und Workflow-Worker nun Abfragen durchführen, können Sie die Workflow-Ausführung starten. Dieser Prozess läuft so lange, bis der Workflow den Status "abgeschlossen" zurückgibt. Führen Sie ihn in einem neuen Terminal-Fenster aus (außer Sie haben die Worker mit dem &-Operator in ihre eigenen separaten Prozesse abzweigen lassen).

```
fi
```

> **Note**
>
> Wenn Sie eigene Eingabedaten angeben möchten, die zuerst an den Workflow und dann an die Aktivität übergeben werden, fügen Sie sie zur Befehlszeile hinzu. Beispiel:

```
echo "## Running $className..."
```

Sobald Sie die Workflow-Ausführung starten, sollten Sie Ausgaben von beiden Workern und von der Workflow-Ausführung selbst sehen. Wenn der Workflow schließlich abgeschlossen ist, wird die Ausgabe auf dem Bildschirm angezeigt.

Vollständiger Quellcode für dieses Beispiel

Sie können den vollständigen Quellcode für dieses Beispiel auf Github im aws-java-developer-guideRepository durchsuchen.

Weitere Informationen

- Die hier gezeigten Worker können zu verloren gegangenen Aufgaben führen, wenn sie beendet werden, während noch eine Workflow-Abfrage läuft. Unter Korrektes Herunterfahren von Aktivitäts- und Workflow-Workern erfahren Sie, wie sich Worker korrekt herunterfahren lassen.
- Weitere Informationen Amazon SWF finden Sie auf der Amazon SWFHomepage oder im Amazon SWFEntwicklerleitfaden.
- Sie können das AWS Flow Framework for Java verwenden, um komplexere Workflows in einem eleganten Java-Stil mithilfe von Anmerkungen zu schreiben. Weitere Informationen finden Sie im Entwicklerhandbuch AWS Flow Framework für den Befehl sieht dann wie aus.

Lambda-Aufgaben

Als Alternative zu oder in Verbindung mit Amazon SWF-Aktivitäten können Sie Lambda-Funktionen nutzen, um Arbeitseinheiten in Ihren Workflows darzustellen, und sie ähnlich wie Aktivitäten planen.

In diesem Thema wird hauptsächlich auf die Implementierung eingegangenAmazon SWF Lambda-Aufgaben mit demAWS SDK for Javaaus. Weitere Informationen zuLambdaAufgaben im Allgemeinen sieheAWS LambdaAufgabenimAmazon SWFEntwicklerhandbuch.

Einrichten einer serviceübergreifenden IAM-Rolle zum Ausführen Ihrer Lambda-Funktion

VorAmazon SWFkann dein laufenLambda-Funktion müssen Sie eine IAM-Rolle einrichten, umAmazon SWFBerechtigung zum AusführenLambda-Funktionen in Ihrem Auftrag. Weitere Informationen über die folgende Vorgehensweise finden Sie unterAWS LambdaAufgabenaus.

Sie benötigen den Amazon Resource Name (ARN) dieser IAM-Rolle bei der Registrierung eines Workflows, der verwendetLambda-Aufgaben.

Erstellen einer Lambda-Funktion

Sie können Lambda-Funktionen in verschiedenen Sprachen verfassen, einschließlich Java. Weitere Informationen zum Erstellen, Bereitstellen und VerwendenLambda-Funktionen finden Sie imAWS LambdaEntwicklerhandbuchaus.

> **Note**
>
> Es spielt keine Rolle, in welcher Sprache Sie zum Schreiben IhrerLambdaFunktion, kann es geplant und ausgeführt werden vonirgendein Amazon SWF-Workflow unabhängig von der Sprache, in der Ihr Workflow-Code geschrieben wurde.Amazon SWFübernimmt die Details der Ausführung dieser Funktion sowie die Übergabe von Daten in die und aus der Funktion.

Hier ist ein EinfachesLambda-Funktion, die anstelle der Aktivität in verwendet werden könnteErstellen eines einfachenAmazon SWFAnwendungaus.

- Diese Version wurde in JavaScript geschrieben und kann direkt mit demAWS Management Console:

```
exports.handler = function(event, context) {
    context.succeed("Hello, " + event.who + "!");
};
```

- Hier ist die gleiche Funktion in Java geschrieben. Diese könnten Sie ebenfalls auf Lambda bereitstellen und ausführen:

```
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**
>
> Weitere Informationen zum Bereitstellen von Java-Funktionen auf Lambda finden Sie unterErstellen eines Bereitstellungspakets (Java)imAWS LambdaEntwicklerhandbuch. Sie können auch den Abschnitt mit dem Titel ansehenProgrammiermodell für die ErstellungLambda-Funktionen in Javaaus.

Lambda-Funktionen nehmen ein Ereignis- oder Eingabeobjekt als ersten Parameter und ein Kontextobjekt als zweiten Parameter entgegen. Letzteres bietet Informationen über die Anforderung zur Ausführung der Lambda-Funktion. Diese bestimmte Funktion erwartet die Parameter im JSON-Format, wobei das Feld `who` auf den Namen zum Zusammensetzen der Begrüßung gesetzt ist.

Registrieren eines Workflows zur Nutzung mit Lambda

Damit ein Workflow eine Lambda-Funktion planen kann, sollten Sie den Namen der IAM-Rolle angeben, durch die Amazon SWF die Berechtigung zum Aufrufen von Lambda-Funktionen erhält. Sie können die Rolle während der Registrierung des Workflows festlegen, indem Sie die Methode

`withDefaultLambdaRole` oder `setDefaultLambdaRole` im RegisterWorkflowTypeRequest nutzen.

```
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) {
```

Planen einer Lambda-Aufgabe

Lambda-Aufgaben können ähnlich wie Aktivitäten geplant werden. Sie müssen einenEntscheidungmit einer `ScheduleLambdaFunction`DecisionTypeund mitScheduleLambdaFunctionDecisionAttributesaus.

```
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(
```

In den `ScheduleLambdaFuntionDecisionAttributes` sollten Sie einen Namen angeben, also den ARN der Lambda-Funktion, die aufgerufen werden soll. Außerdem wird eine id benötigt, wobei es

sich um den Namen handelt, mit dem Amazon SWF die Lambda-Funktion in den Verlaufsprotokollen identifiziert.

Optional können Sie der -Funktion auch einen inputLambda angeben und ihren Wert für die maximale Ausführungszeit festlegen. Dabei handelt es sich um die Anzahl an Sekunden, die die Lambda-Funktion ausgeführt werden darf, bevor ein `LambdaFunctionTimedOut`-Ereignis ausgelöst wird.

> **Note**
>
> Dieser Code nutzt den AWSLambdaClient zum Abrufen des ARNs der Lambda-Funktion über ihren Funktionsnamen. Mit dieser Technik können Sie vermeiden, den vollen ARN (der IhreAWS-KontoID) in Ihrem Code.

Umgang mit Lambda-Funktionsereignissen in Ihrem Entscheider

Lambda-Aufgaben generieren eine Reihe von Ereignissen, auf die Sie beim Abrufen von Entscheidungsaufgaben in Ihrem Workflow-Worker je nach Lebenszyklus der Lambda-Aufgabe mit EventType-Werten wie `LambdaFunctionScheduled`, `LambdaFunctionStarted` und `LambdaFunctionCompleted` reagieren können. Wenn die Lambda-Funktion fehlschlägt oder länger als ihr festgelegter Timeout-Wert ausgeführt wird, erhalten Sie jeweils den Ereignistyp `LambdaFunctionFailed` oder `LambdaFunctionTimedOut`.

```
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;
```

Erhalten von Ausgaben aus der Lambda-Funktion

Wenn Sie ein`LambdaFunctionCompleted`[`EventType`], you can retrieve your 0 function's return value by first calling `getLambdaFunctionCompletedEventAttributesauf derhistoryEventum einen zu bekommenlambdaFunctionCompletedEventAttributesObjekt, und ruft dann seinegetResultMethode zum Abrufen der Ausgabe desLambda-Funktion:

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

Vollständiger Quellcode für dieses Beispiel

Sie können den vollständigen Quellcode :github:`<awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf_lambda/> für dieses Beispiel auf Github im Repository aws-java-developer-guide durchsuchen.

Korrektes Herunterfahren von Aktivitäts- und Workflow-Workern

DieErstellen einer einfachenAmazon SWFAnwendung-Thema bot eine vollständige Implementierung einer einfachen Workflow-Anwendung, die aus einer Registrierungsanwendung, einem Aktivitäts- und Workflow-Worker und einem Workflow-Starter besteht.

Worker-Klassen sind für den fortlaufenden Betrieb konzipiert und fragen stetig Aufgaben ab, die von Amazon SWF gesendet werden, um Aktivitäten auszuführen oder Entscheidungen zurückgeben zu können. Sobald eine Abfrage-Anforderung ausgeführt wird, nimmt Amazon SWF Notiz vom anfragenden Worker und versucht, diesem eine Aufgabe zuzuweisen.

Wenn der Workflow-Worker während einer langen Abfrage beendet wird, versucht Amazon SWF eventuell weiterhin, eine Aufgabe an den beendeten Worker zu senden. Dies kann zu einer verlorenen Aufgabe führen (bis die Aufgabe abläuft).

Eine Möglichkeit zur Bewältigung dieser Situation besteht darin, zu warten, bis alle Long-Poll-Anforderungen zurückkehren, bevor der Worker beendet wird.

In diesem Thema schreiben wir den Aktivitäts-Worker von `helloswf` um und verwenden Java-Hooks für das Herunterfahren. So versuchen wir, den Aktivitäts-Worker ordnungsgemäß herunterzufahren.

Hier finden Sie den vollständigen Code:

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

In dieser Version wurde der Polling-Code aus der `main`-Funktion in der ursprünglichen Version in eine eigene Methode `pollAndExecute` verschoben.

Die `main`-Funktion nutzt jetzt einen CountDownLatch zusammen mit einem Shutdown-Hook. Dies bewirkt, dass der Thread bis zu 60 Sekunden nach der angeforderten Beendigung wartet, bevor der Thread heruntergefahren wird.

Registrieren von Domänen

Jeder Workflow und jede Aktivität inAmazon SWFbraucht einDomäneum hereinzulaufen.

1. Erstellen Sie ein neues RegisterDomainRequest-Objekt und übergeben Sie ihm mindestens einen Domänennamen und einen Aufbewahrungszeitraum für die Workflow-Ausführung (beide Parameter sind erforderlich).
2. Rufen Sie die Methode AmazonSimpleWorkflowClient.registerDomain mit dem RegisterDomainRequest-Objekt auf.

3. Fangen Sie die DomainAlreadyExistsException ab, die ausgelöst wird, falls die angeforderte Domäne bereits vorhanden ist (in diesem Fall ist gewöhnlich keine Aktion erforderlich).

Der folgende Code veranschaulicht dieses Vorgehen:

```
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!");
    }
}
```

Auflisten von Domänen

Sie können dasAmazon SWFDomänen, die dem -Konto zugeordnet sindAWSRegion nach Registrierungsart.

1. Erstellen einesListDomainsRequest-Objekt und geben Sie den Registrierungsstatus der Domänen an, an denen Sie interessiert sind. Diese Angabe ist erforderlich.
2. Rufen Sie AmazonSimpleWorkflowClient.listDomains mit dem ListDomainRequest-Objekt auf. Die Ergebnisse werden in einem DomainInfos-Objekt bereitgestellt.
3. Rufen Sie getDomainInfos auf dem zurückgegebenen Objekt auf und Sie erhalten eine Liste mit DomainInfo-Objekten.
4. Rufen Sie getname für jedes DomainInfo-Objekt auf, um dessen Namen zu erhalten.

Der folgende Code veranschaulicht dieses Vorgehen:

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

Codebeispiele, die im SDK enthalten sind

Das AWS SDK for Java enthält Codebeispiele, die viele der Funktionen des SDKs als erstellbare, ausführbare Programme zeigen. Sie können diese Beispiele untersuchen oder ändern und so Ihre eigenen implementierenAWSLösungen mit demAWS SDK for Javaaus.

Abrufen der Beispiele

Die AWS SDK for Java-Codebeispiele befinden sich im Verzeichnis samples des SDKs. Wenn Sie das SDK mit den Informationen in heruntergeladen und installiert habenEinrichten derAWS SDK for Javahaben Sie bereits die Samples auf Ihrem System.

Sie können die neuesten Beispiele auch im AWS SDK for Java GitHub-Repository ansehen, und zwar im Verzeichnis src/samples.

Erstellen und Ausführen der Beispiele in der Befehlszeile

Mithilfe der enthaltenen Ant-Build-Skripts können Sie die Beispiele leicht erstellen und in der Befehlszeile ausführen. Jedes Beispiel enthält eine README-Datei im HTML-Format mit speziellen Informationen für das jeweilige Beispiel.

> **Note**
>
> Wenn Sie den Beispiel-Code auf GitHub ansehen, öffnen Sie die README.html-Datei des Beispiels und klicken Sie auf die Schaltfläche Raw in der Quellcode-Ansicht. Im Raw-Modus wird das HTML korrekt im Browser dargestellt.

Voraussetzungen

Bevor Sie einen derAWS SDK for JavaSamples, müssen Sie Ihre einstellenAWS-Anmeldeinformationen in der Umgebung oder mit demAWS CLI, wie in

angegebenEinrichtenAWSAnmeldeinformationen und Region für die Entwicklungaus. Die Beispiele verwenden die standardmäßige Anbieterkette von Anmeldeinformationen, sofern möglich. Indem Sie also die Anmeldeinformationen also so einrichten, können Sie die riskante Praxis vermeiden, die einzufügenAWS-Anmeldeinformationen in Dateien innerhalb des Quellcodeverzeichnisses (wo sie versehentlich eingecheckt und offengelegt werden könnten).

Ausführen der Beispiele

1. Wechseln Sie in das Verzeichnis mit dem Beispiel-Code. Angenommen, Sie befinden sich im Stammverzeichnis desAWSSDK-Download und möchte das`AwsConsoleApp`Beispiel, geben Sie den folgenden Befehl ein:

   ```
   cd samples/AwsConsoleApp
   ```

2. Erstellen und führen Sie das Beispiel mit Ant aus. Das Standard-Build-Ziel führt beide Aktionen aus, daher können Sie einfach Folgendes eingeben:

   ```
   ant
   ```

Das Beispiel gibt Informationen an die Standardausgabe aus, z. B.:

```
===========================================

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.
```

Erstellen und Ausführen der Beispiele in der Eclipse-IDE

Wenn Sie den AWS Toolkit for Eclipse verwenden, können Sie auch in Eclipse ein neues Projekt basierend auf dem AWS SDK for Java starten oder das SDK zu einem bereits vorhandenen Java-Projekt hinzufügen.

Voraussetzungen

Nach der Installation des AWS Toolkit for Eclipse empfehlen wir das Konfigurieren des Toolkits mit Ihren Anmeldeinformationen. Sie können dies jederzeit tun, indem SiePräferenzenausWindowMenü in Eclipse, und dann wählen Sie die AWSToolkitAbschnitts erstellt.

Ausführen der Beispiele

1. Öffnen Sie Eclipse.
2. Erstellen eines neuenAWSJava-Projekt. Klicken Sie in Eclipse im Menü Datei auf Neu und dann auf Projekt. Der Assistent Neues Projekt wird geöffnet.
3. Erweitern Sie den AWS Kategorie, dann wähle AWSJava-Projektaus.
4. Wählen Sie Next (Weiter) aus. Die Seite "Projekteinstellungen" wird angezeigt.
5. Geben Sie einen Namen in das Feld Projektname ein. DieAWS SDK for JavaIn der Gruppe „Beispiele“ werden die im SDK verfügbaren Beispiele angezeigt, wie zuvor beschrieben.
6. Wählen Sie durch Markieren der Kontrollkästchen die Beispiele aus, die Sie in Ihr Projekt aufnehmen möchten.
7. Geben Sie IhrenAWS-Anmeldeinformationen. Wenn Sie das AWS Toolkit for Eclipse bereits mit Ihren Anmeldeinformationen konfiguriert haben, werden diese automatisch eingetragen.
8. Klicken Sie auf Finish (Abschließen). Das Projekt wird erstellt und zum Projekt-Explorer hinzugefügt.
9. Wählen Sie die `.java`-Beispieldatei aus, die Sie ausführen möchten. Für das Amazon S3-Beispiel wählen Sie z. B. `S3Sample.java` aus.
10. Klicken Sie im Menü Ausführen auf Ausführen.
11. Klicken Sie im Projekt-Explorer mit der rechten Maustaste auf das Projekt, zeigen Sie dann auf Build-Pfad und wählen Sie Bibliotheken hinzufügen.
12. Klicken Sie auf AWSJava-SDK, wählenWeiterund folgen Sie dann den Anweisungen auf dem Bildschirm.

Sicherheit für die AWS SDK for Java

Cloud-Sicherheit genießt bei Amazon Web Services (AWS) höchste Priorität. Als AWS -Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die zur Erfüllung der Anforderungen von Organisationen entwickelt wurden, für die Sicherheit eine kritische Bedeutung hat. Sicherheit ist eine geteilte Verantwortung zwischen AWS und Ihnen. Im Modell der übergreifenden Verantwortlichkeit wird Folgendes mit „Sicherheit der Cloud“ bzw. „Sicherheit in der Cloud“ umschrieben:

Sicherheit der Cloud – AWS ist verantwortlich für den Schutz der Infrastruktur, die alle in der AWS Cloud angebotenen Services ausführt, und für die Bereitstellung von Services, die Sie sicher nutzen können. Unsere Sicherheitsverantwortung hat bei höchste Priorität AWS, und die Effektivität unserer Sicherheit wird regelmäßig von externen Prüfern im Rahmen der AWS -Compliance-Programme getestet und überprüft.

Sicherheit in der Cloud – Ihre Verantwortung wird durch den AWS Service bestimmt, den Sie verwenden, sowie durch andere Faktoren wie die Vertraulichkeit Ihrer Daten, die Anforderungen Ihrer Organisation und die geltenden Gesetze und Vorschriften.

Dieses AWS Produkt oder dieser Service folgt dem Modell der geteilten Verantwortung durch die spezifischen Amazon Web Services (AWS)-Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der AWS Seite zur Servicesicherheit und AWS in den Services, die im Rahmen von AWS Compliance-Bemühungen nach Compliance-Programmen enthalten sind.

Themen

Datenschutz in AWS SDK for Java 1.x

Das -Modell der geteilten Verantwortung gilt für den Datenschutz in diesem AWS Produkt oder Service. Wie in diesem Modell beschrieben, AWS ist für den Schutz der globalen Infrastruktur verantwortlich, die die gesamte AWS Cloud betreibt. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt enthält die Sicherheitskonfigurations- und Verwaltungsaufgaben für die von Ihnen verwendeten AWS -Services. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS Modell der geteilten Verantwortung und zur DSGVO im - AWS Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, die AWS-Konto Anmeldeinformationen zu schützen und individuelle Benutzerkonten mit AWS Identity and Access Management () einzurichten IAM. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem sollten Sie die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
- Verwenden Sie SSL/TLS für die Kommunikation mit - AWS Ressourcen.
- Richten Sie die API- und Benutzeraktivitätsprotokollierung mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen mit allen Standardsicherheitskontrollen innerhalb von - AWS Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die bei der Erkennung und Sicherung personenbezogener Daten helfen, die in gespeichert sind Amazon S3.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder eine API FIPS-140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern wie z. B. im Feld Name keine sensiblen, identifizierenden Informationen wie Kontonummern von Kunden einzugeben. Dies gilt auch, wenn Sie mit diesem AWS Produkt oder Service oder anderen - AWS Services unter Verwendung der Konsole, API AWS CLI oder AWS SDKs arbeiten. Alle Daten, die Sie in dieses AWS Produkt oder diesen Service oder andere Services eingeben, können in Diagnoseprotokolle aufgenommen werden. Wenn Sie eine URL für einen externen Server bereitstellen, schließen Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL ein.

AWS SDK for Java -Unterstützung für TLS

Die folgenden Informationen gelten nur für die Java-SSL-Implementierung (die Standard-SSL-Implementierung in der AWS SDK for Java). Wenn Sie eine andere SSL-Implementierung verwenden, erfahren Sie in Ihrer spezifischen SSL-Implementierung, wie Sie TLS-Versionen erzwingen.

Vorgehensweise zum Überprüfen der TLS-Version

Lesen Sie die Dokumentation Ihres Java Virtual Machine (JVM)-Anbieters, um festzustellen, welche TLS-Versionen auf Ihrer Plattform unterstützt werden. Für einige JVMs wird mit dem folgenden Code gedruckt, welche SSL-Versionen unterstützt werden.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProto
```

Um den SSL-Handshake in Aktion zu sehen und welche Version von TLS verwendet wird, können Sie die Systemeigenschaft javax.net.debug verwenden.

```
java app.jar -Djavax.net.debug=ssl
```

> **Note**
>
> TLS 1.3 ist mit den SDK for Java-Versionen 1.9.5 bis 1.10.31 nicht kompatibel. Weitere Informationen finden Sie im folgenden Blogbeitrag. https://aws.amazon.com/blogs/developer/tls-1-3-incompatibility-with-aws-sdkfor-java-versions-1-9-5-to-1-10-31/

Erzwingen einer Mindest-TLS-Version

Das SDK bevorzugt immer die neueste TLS-Version, die von der Plattform und dem Service unterstützt wird. Wenn Sie eine bestimmte TLS-Mindestversion erzwingen möchten, lesen Sie die Dokumentation Ihrer JVM. Für OpenJDK-basierte JVMs können Sie die Systemeigenschaft verwenden`jdk.tls.client.protocols`.

```
java app.jar -Djdk.tls.client.protocols=PROTOCOLS
```

Die unterstützten Werte von PROTOCOLS finden Sie in der Dokumentation Ihrer JVM.

Identitäts- und Zugriffsverwaltung

AWS Identity and Access Management (IAM) ist ein AWS-Service , mit dem ein Administrator den Zugriff auf - AWS Ressourcen sicher steuern kann. IAM-Administratoren steuern, wer für die Nutzung von - AWS Ressourcen authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) werden kann. IAM ist ein AWS-Service , den Sie ohne zusätzliche Kosten verwenden können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in AWS.

Service-Benutzer – Wenn Sie AWS-Services zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie für Ihre Arbeit weitere AWS Funktionen ausführen, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie nicht auf ein Feature in zugreifen können AWS, finden Sie weitere Informationen unter Fehlerbehebung für AWS Identität und Zugriff oder im Benutzerhandbuch des von AWS-Service Ihnen verwendeten .

Service-Administrator – Wenn Sie in Ihrem Unternehmen für AWS Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollständigen Zugriff auf AWS. Es ist Ihre Aufgabe, zu bestimmen, auf welche AWS Funktionen und Ressourcen Ihre Service-Benutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM mit verwenden kann AWS, finden Sie im Benutzerhandbuch der , die AWS-Service Sie verwenden.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf AWS verfassen können. Beispiele für AWS

identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie im Benutzerhandbuch der , die AWS-Service Sie verwenden.

Authentifizierung mit Identitäten

Die Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten bei anmelden. Sie müssen als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle authentifiziert (bei angemeldet AWS) sein.

Sie können sich bei AWS als Verbundidentität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt werden. AWS IAM Identity Center (IAM Identity Center)-Benutzer, die Single-Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für Verbundidentitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie AWS über einen Verbund auf zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, um welchen Benutzertyp es sich handelt, können Sie sich bei der AWS Management Console oder im - AWS Zugriffsportal anmelden. Weitere Informationen zur Anmeldung bei AWS finden Sie unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung - Benutzerhandbuch.

Wenn Sie AWS programmgesteuert auf zugreifen, AWS stellt ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (Command Line Interface, CLI) bereit, um Ihre Anforderungen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenständigen Signieren von Anforderungen finden Sie unter Signieren von AWS API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. empfiehlt beispielsweise, AWS Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie ein erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen im Konto hat. Diese Identität wird als AWS-Konto

Root-Benutzer bezeichnet und Sie melden sich mit der E-Mail-Adresse und dem Passwort an, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Fordern Sie als bewährte Methode menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, auf, den Verbund mit einem Identitätsanbieter zu verwenden, um AWS-Services mithilfe temporärer Anmeldeinformationen auf zuzugreifen.

Eine Verbundidentität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, ein Web-Identitätsanbieter, die AWS Directory Service, das Identity-Center-Verzeichnis oder jeder Benutzer, der mit AWS-Services Anmeldeinformationen auf zugreift, die über eine Identitätsquelle bereitgestellt werden. Wenn Verbundidentitäten auf zugreifen AWS-Konten, übernehmen sie Rollen und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen oder eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und synchronisieren, um sie für alle Ihre AWS-Konten und Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche

Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI - oder AWS -API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können AWS-Services Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden

Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

- Serviceübergreifender Zugriff – Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen in auszuführen AWS, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung AWS-Service , Anfragen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Service eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder -Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle: Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
  - Serviceverknüpfte Rolle – Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einem verknüpft ist AWS-Service. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem angezeigt AWS-Konto und gehören dem Service. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.
- Anwendungen, die auf Amazon EC2 ausgeführt werden – Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und - AWS CLI oder AWS -API-Anforderungen stellen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine - AWS Rolle zuzuweisen und sie für alle ihre Anwendungen verfügbar zu machen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie steuern den Zugriff in , AWS indem Sie Richtlinien erstellen und sie an AWS Identitäten oder Ressourcen anfügen. Eine Richtlinie ist ein Objekt in , AWS das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können AWS JSON-Richtlinien verwenden, um festzulegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen aus der AWS Management Console, der AWS CLI oder der AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die

Sie mehreren Benutzern, Gruppen und Rollen in Ihrem anfügen können AWS-Konto. Verwaltete Richtlinien umfassen - AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder umfassen AWS-Services.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services AWS WAF, die ACLs unterstützen. Weitere Informationen zu ACLs finden Sie unter Zugriffssteuerungsliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze

für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.

- Service-Kontrollrichtlinien (SCPs)– SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in angeben AWS Organizations. AWS Organizations ist ein Service zum Gruppieren und zentralen Verwalten mehrerer AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Die SCP beschränkt Berechtigungen für Entitäten in Mitgliedskonten, einschließlich jeder Root-Benutzer des AWS-Kontos. Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS bestimmt, ob eine Anforderung zugelassen werden soll, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter Logik zur Richtlinienbewertung im IAM-Benutzerhandbuch.

Funktionsweise AWS-Services von mit IAM

Einen Überblick über die AWS-Services Funktionsweise der meisten IAM-Funktionen finden Sie unter -AWS Services, die mit IAM funktionieren im IAM-Benutzerhandbuch.

Informationen zur Verwendung eines bestimmten AWS-Service mit IAM finden Sie im Abschnitt Sicherheit im Benutzerhandbuch des entsprechenden Services.

Fehlerbehebung für AWS Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufige Probleme zu diagnostizieren und zu beheben, die beim Arbeiten mit AWS und IAM auftreten können.

Themen

Ich bin nicht autorisiert, eine Aktion in auszuführen AWS

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `awes:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `awes:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht autorisiert, iam durchzuführen:PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an AWSübergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine vorhandene Rolle an diesen Service zu übergeben, anstatt eine neue Servicerolle oder serviceverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

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

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine - AWS Ressourcen gewähren

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Services, die ressourcenbasierte Richtlinien oder Zugriffssteuerungslisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob diese Funktionen AWS unterstützt, finden Sie unter Funktionsweise AWS-Services von mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre AWS-Konten -Ressourcen in Ihrem Besitz finden Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto , das Sie besitzen im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre -Ressourcen gewähren AWS-Konten, finden Sie unter Gewähren des Zugriffs auf AWS-Konten von Dritten im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

Compliance-Validierung für dieses AWS Produkt oder diesen Service

Informationen darüber, ob ein in den Geltungsbereich bestimmter Compliance-Programme AWS-Service fällt, finden Sie AWS-Services unter im Geltungsbereich nach Compliance-Programm und wählen Sie das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme

Sie können Auditberichte von Drittanbietern mit herunterladen AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten unter AWS Artifact .

Ihre Compliance-Verantwortung bei der Verwendung von AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Unterstützung der Compliance bereit:

- Schnellstartanleitungen für Sicherheit und Compliance – In diesen Bereitstellungsleitfäden werden Überlegungen zur Architektur erörtert und Schritte für die Bereitstellung von Basisumgebungen in bereitgestellt AWS , die sich auf Sicherheit und Compliance konzentrieren.
- Architekturerstellung für HIPAA-Sicherheit und -Compliance in Amazon Web Services – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe AWS von HIPAA-berechtigte Anwendungen erstellen können.

  > **Note**
  >
  > Nicht alle AWS-Services sind HIPAA-berechtigt. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen – Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.
- AWS Kunden-Compliance-Leitfäden – Verstehen Sie das Modell der geteilten Verantwortung anhand der Berücksichtigung der Compliance. Die Leitfäden fassen die bewährten Methoden zur Sicherung zusammen AWS-Services und ordnen die Leitlinien den Sicherheitskontrollen in mehreren Frameworks zu (einschließlich National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Officer (PCI) und International Organization for Standardization (ISO)).

- Bewertung von Ressourcen mit Regeln im -AWS Config Entwicklerhandbuch – Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub – Dies AWS-Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus innerhalb von AWS. Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- AWS Audit Manager – Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um den Umgang mit Risiken und die Einhaltung von Branchenstandards zu vereinfachen.

Dieses AWS Produkt oder dieser Service folgt dem Modell der geteilten Verantwortung über die spezifischen Amazon Web Services (AWS)-Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der AWS Seite zur Servicesicherheit und AWS in den Services, die im Rahmen von AWS Compliance-Bemühungen nach Compliance-Programmen enthalten sind.

Ausfallsicherheit für dieses AWS Produkt oder diesen Service

Die AWS globale -Infrastruktur ist um AWS-Regionen und Availability Zones herum aufgebaut.

AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die mit einem Netzwerk mit niedriger Latenz, hohem Durchsatz und hoher Redundanz verbunden sind.

Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Dieses AWS Produkt oder dieser Service folgt dem Modell der geteilten Verantwortung über die spezifischen Amazon Web Services (AWS)-Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der AWS Seite mit der Service-Sicherheitsdokumentation und AWS den Services, die im Rahmen der AWS Compliance-Bemühungen nach Compliance-Programmen enthalten sind.

Infrastruktursicherheit für dieses AWS Produkt oder diesen Service

Dieses AWS Produkt oder dieser Service verwendet verwaltete Services und ist daher durch die AWS globale Netzwerksicherheit von geschützt. Informationen zu AWS Sicherheitsservices und wie die Infrastruktur AWS schützt, finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung mit den bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastrukturschutz in Security Pillar AWS Well-Architected Framework.

Sie verwenden durch AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf dieses AWS Produkt oder diesen Service zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Dieses AWS Produkt oder dieser Service folgt dem Modell der geteilten Verantwortung über die spezifischen Amazon Web Services (AWS)-Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der AWS Seite zur Servicesicherheit und AWS in den Services, die im Rahmen von AWS Compliance-Bemühungen nach Compliance-Programmen enthalten sind.

Amazon S3 Migration des Verschlüsselungs-Clients

In diesem Thema erfahren Sie, wie Sie Ihre Anwendungen von Version 1 (V1) des Amazon Simple Storage Service (Amazon S3)-Verschlüsselungsclients zu Version 2 (V2) migrieren und die Anwendungsverfügbarkeit während des gesamten Migrationsprozesses sicherstellen.

Voraussetzungen

Amazon S3 Die clientseitige Verschlüsselung erfordert Folgendes:

- Java 8 oder höher ist in Ihrer Anwendungsumgebung installiert. Das AWS SDK for Java funktioniert mit dem Oracle Java SE Development Kit und mit Distributionen von Open Java Development Kit (OpenJDK), wie Amazon Corretto, Red Hat OpenJDK und AdoptOpenJDK.
- Das Bolncy Bol Crypto-Paket . Sie können die .jar-Datei in den Klassenpfad Ihrer Anwendungsumgebung platzieren oder Ihrer Maven-`pom.xml`Datei eine Abhängigkeit von der artifactId `bcprov-ext-jdk15on` (mit der groupId `org.bouncycastle`) hinzufügen.

Migrationsübersicht

Diese Migration erfolgt in zwei Phasen:

1. Aktualisieren Sie vorhandene Clients, um neue Formate zu lesen. Aktualisieren Sie Ihre Anwendung, um Version 1.11.837 oder höher von zu verwenden, AWS SDK for Java und stellen Sie die Anwendung erneut bereit. Auf diese Weise können die Amazon S3 Client-seitigen Verschlüsselungsservice-Clients in Ihrer Anwendung Objekte entschlüsseln, die von V2-Service-Clients erstellt wurden. Wenn Ihre Anwendung mehrere AWS SDKs verwendet, müssen Sie jedes SDK separat aktualisieren.
2. Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients zu V2. Sobald alle Ihre V1-Verschlüsselungsclients V2-Verschlüsselungsformate lesen können, aktualisieren Sie die Amazon S3 clientseitigen Verschlüsselungs- und Entschlüsselungsclients in Ihrem Anwendungscode, um ihre V2-Entsprechungen zu verwenden.

Aktualisieren vorhandener Clients zum Lesen neuer Formate

Der V2-Verschlüsselungsclient verwendet Verschlüsselungsalgorithmen, die ältere Versionen von AWS SDK for Java nicht unterstützen.

Der erste Schritt bei der Migration besteht darin, Ihre V1-Verschlüsselungsclients so zu aktualisieren, dass sie Version 1.11.837 oder höher des verwenden AWS SDK for Java. (Es wird empfohlen, auf die neueste Version zu aktualisieren, die Sie in der Java-API-Referenzversion 1.x finden.) Aktualisieren Sie dazu die Abhängigkeit in Ihrer Projektkonfiguration. Nachdem Ihre Projektkonfiguration aktualisiert wurde, erstellen Sie Ihr Projekt neu und stellen Sie es erneut bereit.

Sobald Sie diese Schritte abgeschlossen haben, können die V1-Verschlüsselungsclients Ihrer Anwendung Objekte lesen, die von V2-Verschlüsselungsclients geschrieben wurden.

Aktualisieren der Abhängigkeit in Ihrer Projektkonfiguration

Ändern Sie Ihre Projektkonfigurationsdatei (z. B. pom.xml oder build.gradle), um Version 1.11.837 oder höher des zu verwenden AWS SDK for Java. Erstellen Sie dann Ihr Projekt neu und stellen Sie es erneut bereit.

Wenn Sie diesen Schritt ausführen, bevor Sie neuen Anwendungscode bereitstellen, können Sie sicherstellen, dass Verschlüsselungs- und Entschlüsselungsvorgänge während des Migrationsprozesses in Ihrer gesamten Flotte konsistent bleiben.

Beispiel für die Verwendung von Maven

Ausschnitt aus einer pom.xml-Datei:

```
<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>
```

Beispiel für die Verwendung von Gradle

Ausschnitt aus einer build.gradle-Datei:

```
dependencies {
  implementation platform('com.amazonaws:aws-java-sdk-bom:1.11.837')
  implementation 'com.amazonaws:aws-java-sdk-s3'
}
```

Migrieren von Verschlüsselungs- und Entschlüsselungsclients zu V2

Sobald Ihr Projekt mit der neuesten SDK-Version aktualisiert wurde, können Sie Ihren Anwendungscode ändern, um den V2-Client zu verwenden. Aktualisieren Sie dazu zunächst Ihren Code, um den neuen Service-Client-Builder zu verwenden. Stellen Sie dann

Verschlüsselungsmaterialien mit einer Methode im Builder bereit, die umbenannt wurde, und konfigurieren Sie Ihren Service-Client nach Bedarf weiter.

Diese Codeausschnitte veranschaulichen, wie die clientseitige Verschlüsselung mit der verwendet wird AWS SDK for Java, und bieten Vergleiche zwischen den Verschlüsselungsclients V1 und 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();
```

Im obigen Beispiel `cryptoMode` wird auf festgelegt`AuthenticatedEncryption`. Dies ist eine Einstellung, die es einem V2-Verschlüsselungsclient ermöglicht, Objekte zu lesen, die von einem V1-Verschlüsselungsclient geschrieben wurden. Wenn Ihr Client nicht die Fähigkeit zum Lesen von Objekten benötigt, die von einem V1-Client geschrieben wurden, empfehlen wir `StrictAuthenticatedEncryption` stattdessen, die Standardeinstellung von zu verwenden.

Erstellen eines V2-Verschlüsselungs-Clients

Der V2-Verschlüsselungsclient kann durch Aufrufen von AmazonS3EncryptionClientV2.encryptionBuilder () erstellt werden.

Sie können alle Ihre vorhandenen V1-Verschlüsselungsclients durch V2-Verschlüsselungsclients ersetzen. Ein V2-Verschlüsselungsclient kann jedes Objekt, das von einem V1-Verschlüsselungsclient geschrieben wurde, immer lesen, solange Sie ihm dies

erlauben, indem Sie den V2-Verschlüsselungsclient für die Verwendung des `konfigurierenAuthenticatedEncryption`cryptoMode.

Das Erstellen eines neuen V2-Verschlüsselungsclients ähnelt dem Erstellen eines V1-Verschlüsselungsclients. Es gibt jedoch einige Unterschiede:

- Sie verwenden ein -CryptoConfigurationV2Objekt, um den Client anstelle eines -CryptoConfigurationObjekts zu konfigurieren. Dieser Parameter muss angegeben werden.
- Die cryptoMode Standardeinstellung für den V2-Verschlüsselungsclient ist StrictAuthenticatedEncryption. Für den V1-Verschlüsselungsclient lautet er EncryptionOnly.
- Die Methode withEncryptionMaterials() im Verschlüsselungs-Client-Builder wurde in withEncryptionMaterialsAnbieter () umbenannt. Dies ist nur eine Telefonieänderung, die den Argumenttyp genauer widerspiegelt. Sie müssen die neue Methode verwenden, wenn Sie Ihren Service-Client konfigurieren.

> **Note**
>
> Lesen Sie beim Entschlüsseln mit AES-GCM das gesamte Objekt bis zum Ende, bevor Sie die entschlüsselten Daten verwenden. Dadurch wird überprüft, ob das Objekt seit der Verschlüsselung nicht geändert wurde.

Verwenden von Verschlüsselungsmaterialanbietern

Sie können weiterhin dieselben Verschlüsselungsmaterialanbieter und Verschlüsselungsmaterialobjekte verwenden, die Sie bereits mit dem V1-Verschlüsselungsclient verwenden. Diese Klassen sind dafür verantwortlich, die Schlüssel bereitzustellen, die der Verschlüsselungsclient zum Schutz Ihrer Daten verwendet. Sie können sowohl mit dem V2- als auch mit dem V1-Verschlüsselungsclient austauschbar verwendet werden.

Konfigurieren des V2 Encryption Client

Der V2-Verschlüsselungsclient ist mit einem -CryptoConfigurationV2Objekt konfiguriert. Dieses Objekt kann erstellt werden, indem der Standardkonstruktor aufgerufen und dann seine Eigenschaften nach Bedarf in den Standardwerten geändert werden.

Die Standardwerte für CryptoConfigurationV2 sind:

- `cryptoMode` = `CryptoMode.StrictAuthenticatedEncryption`
- `storageMode` = `CryptoStorageMode.ObjectMetadata`
- `secureRandom` = Instance von `SecureRandom`
- `rangeGetMode` = `CryptoRangeGetMode.DISABLED`
- `unsafeUndecryptableObjectPassthrough` = `false`

Beachten Sie, dass im V2-`cryptoMode`Verschlüsselungsclient nicht unterstützt EncryptionOnly wird. Der V2-Verschlüsselungsclient verschlüsselt Inhalte immer mit authentifizierter Verschlüsselung und schützt Inhaltsverschlüsselungsschlüssel (CEKs) mit V2-`KeyWrap`Objekten.

Das folgende Beispiel zeigt, wie Sie die Crypto-Konfiguration in V1 angeben und wie Sie ein CryptoConfigurationV2-Objekt instanziieren, das an den V2-Verschlüsselungs-Client-Builder übergeben wird.

V1

```
CryptoConfiguration cryptoConfiguration = new CryptoConfiguration()
        .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

V2

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
        .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

Weitere Beispiele

Die folgenden Beispiele zeigen, wie Sie bestimmte Anwendungsfälle im Zusammenhang mit einer Migration von V1 zu V2 angehen können.

Konfigurieren eines Service-Clients zum Lesen von Objekten, die vom V1-Verschlüsselungs-Client erstellt wurden

Um Objekte zu lesen, die zuvor mit einem V1-Verschlüsselungsclient geschrieben wurden, setzen `cryptoMode` Sie auf `AuthenticatedEncryption`. Der folgende Codeausschnitt zeigt, wie Sie ein Konfigurationsobjekt mit dieser Einstellung erstellen.

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
```

```
        .withCryptoMode(CryptoMode.AuthenticatedEncryption);
```

Konfigurieren eines Service-Clients zum Abrufen von Bytebereichen von Objekten

Um `get` aus einem verschlüsselten S3-Objekt einen Bytebereich erreichen zu können, aktivieren Sie die neue Konfigurationseinstellung `rangeGetMode`. Diese Einstellung ist standardmäßig auf dem V2-Verschlüsselungsclient deaktiviert. Beachten Sie, dass ein Bereich auch bei Aktivierung `get` nur für Objekte funktioniert, die mit Algorithmen verschlüsselt wurden, die von der -`cryptoMode`Einstellung des Clients unterstützt werden. Weitere Informationen finden Sie unter CryptoRangeGetMode in der AWS SDK for Java -API-Referenz.

Wenn Sie die verwenden möchten Amazon S3 TransferManager , um mehrteilige Downloads verschlüsselter Amazon S3 Objekte mit dem V2-Verschlüsselungsclient durchzuführen, müssen Sie zuerst die `rangeGetMode` Einstellung auf dem V2-Verschlüsselungsclient aktivieren.

Der folgende Codeausschnitt zeigt, wie Sie den V2-Client für die Ausführung eines Bereichs-konfigurieren`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);
```

OpenPGP-Schlüssel für den AWS SDK for Java

Alle öffentlich verfügbaren Maven-Artefakte für AWS SDK for Java sind mit dem OpenPGP-Standard signiert. Der öffentliche Schlüssel, den Sie zur Überprüfung der Signatur eines Artefakts benötigen, ist im folgenden Abschnitt verfügbar.

Aktueller Schlüssel

Die folgende Tabelle enthält OpenPGP-Schlüsselinformationen für die aktuellen Versionen des SDK for Java 1.x und SDK for Java 2.x.

Schlüssel-ID	0xAC107B386692DADD
Typ	RSA
Größe	4096/4096
Erstellt	2016-06-30
Läuft ab	2024-10-08
Benutzer-ID	AWSSDKs und Tools < @amazon .com> aws-dr-tools
Schlüssel-Fingerabdruck	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 HINZUFÜGEN

Um den folgenden öffentlichen OpenPGP-Schlüssel für das SDK for Java in die Zwischenablage zu kopieren, wählen Sie das Symbol „Kopieren“ in der oberen rechten Ecke.

```
-----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-----
```

Dokumentverlauf

In diesem Thema werden wichtige Änderungen am AWS SDK for Java Entwicklerhandbuch im Laufe seiner Geschichte beschrieben.

Diese Dokumentation wurde erstellt am: 6. Dezember 2023

12. Januar 2024

Fügen Sie ein Banner hinzu, das das Ende der Unterstützung für AWS SDK for Java v1.x ankündigt.

6. Dezember 2023

- Geben Sie den aktuellen OpenPGP-Schlüssel an.

14. März 2023

- Aktualisierter Leitfaden, angepasst an die bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte IAM-Methoden.

28. Juli 2022

- Es wurde eine Warnung hinzugefügt, dass EC2-Classic am 15. August 2022 eingestellt wird.

22. März 2018

- Die Informationen zum Verwalten von Tomcat-Sitzungen im DynamoDB-Beispiel wurden entfernt, da das Tool nicht mehr unterstützt wird.

2. Nov. 2017

- Kryptografiebeispiele für Amazon S3 Verschlüsselungsclients hinzugefügt, einschließlich neuer Themen: Verwenden Sie clientseitige Verschlüsselung und Amazon S3 clientseitige Verschlüsselung mit verwalteten AWS KMS-Schlüsseln und Amazon S3 clientseitige Verschlüsselung mit Client-Hauptschlüsseln. Amazon S3

14. April 2017

- Der AWS SDK for Java Abschnitt „Amazon S3Beispiele zur Verwendung des Buckets“ wurde aktualisiert, darunter neue Themen: Verwaltung von Amazon S3 Zugriffsberechtigungen für Buckets und Objekte und Konfiguration eines Buckets als Website. Amazon S3

4. April 2017

- Ein neues Thema, Enabling Metrics for the, AWS SDK for Java beschreibt, wie Anwendungs- und SDK-Leistungsmetriken für die AWS SDK for Java generiert werden.

3. April 2017

- Dem AWS SDK for Java Abschnitt „ CloudWatch Beispiele verwenden“ wurden neue CloudWatch Beispiele hinzugefügt: Metriken abrufen aus CloudWatch, Veröffentlichen benutzerdefinierter Metrikdaten, Arbeiten mit CloudWatch Alarmen, Verwenden von Alarmaktionen in CloudWatch und Senden von Ereignissen an CloudWatch

27. März 2017

- Weitere Amazon EC2 Beispiele wurden dem AWS SDK for Java Abschnitt „Amazon EC2Beispiele verwenden“ hinzugefügt: Amazon EC2Instanzen verwalten, Elastic IP-Adressen verwenden in Amazon EC2, Regionen und Availability Zones verwenden, Mit Amazon EC2 Schlüsselpaaren arbeiten und Mit Sicherheitsgruppen arbeiten in Amazon EC2.

21. März 2017

- Neue IAM-Beispiele wurden zu den IAM-Beispielen hinzugefügt. Verwenden Sie den AWS SDK for Java Abschnitt: Verwaltung von IAM-Zugriffsschlüsseln, Verwaltung von IAM-Benutzern, Verwendung von IAM-Kontoaliasen, Arbeiten mit IAM-Richtlinienund Arbeiten mit IAM-Serverzertifikaten

13. März 2017

- Dem Amazon SQS Abschnitt wurden drei neue Themen hinzugefügt: Aktivieren von Long Polling für Amazon SQS Nachrichtenwarteschlangen, Einstellen des Sichtbarkeits-Timeouts in und Verwenden von Warteschlangen fürunzustellbare Nachrichten in. Amazon SQSAmazon SQS

26. Januar 2017

- Es wurden ein neues Amazon S3 Thema, Using TransferManager for Amazon S3 Operations, und neue Best Practices für die AWS Entwicklung hinzugefügt, wobei das AWS SDK for Java Thema im Abschnitt Verwenden des Themas enthalten ist. AWS SDK for Java

16. Januar 2017

- Es wurden ein neues Amazon S3 Thema, Verwaltung des Zugriffs auf Amazon S3 Buckets mithilfe von Bucket-Richtlinien, und zwei neue Amazon SQS Themen, Arbeiten mit Amazon SQS Nachrichtenwarteschlangen und Senden, Empfangen und Löschen von Amazon SQS Nachrichten, hinzugefügt.

16. Dezember 2016

- Es wurden neue Beispielthemen hinzugefügt fürDynamoDB: Arbeiten mit Tabellen in DynamoDB und Arbeiten mit Elementen in. DynamoDB

26. Sept. 2016

- Die Themen im Abschnitt „Erweitert“ wurden in „Verwenden von“ verschobenAWS SDK for Java, da sie für die Verwendung des SDK von zentraler Bedeutung sind.

25. August 2016

- Ein neues Thema, Creating Service Clients, wurde dem Abschnitt Verwenden von hinzugefügt, in dem gezeigt wirdAWS SDK for Java, wie Client Builder verwendet werden können, um die Erstellung von AWS-Service Clients zu vereinfachen.

  Der Abschnitt mit den AWS SDK for JavaCodebeispielen wurde mit neuen Beispielen für S3 aktualisiert, die von einem Repository unterstützt werden GitHub, das den vollständigen Beispielcode enthält.

02. Mai 2016

- Dem AWS SDK for Java Abschnitt Verwenden wurde ein neues Thema, Asynchrone Programmierung, hinzugefügt, in dem beschrieben wird, wie mit asynchronen Client-Methoden gearbeitet wird, die `Future` Objekte zurückgeben oder die eine annehmen. `AsyncHandler`

26. Apr. 2016

- Das Thema SSL-Zertifikatanforderungen wurde entfernt, da es nicht mehr relevant ist. Unterstützung für SHA-1-signierte Zertifikate wurde im Jahr 2015 als veraltet markiert und die Website mit den Test-Skripts wurde entfernt.

14. März 2016

- Ein neues Thema namens Amazon SWFLambda-Aufgaben wurde zum Abschnitt hinzugefügt. Darin geht es um die Implementierung eines Amazon SWF-Workflows, der Lambda-Funktionen als Aufgaben aufruft und sich als Alternative zur Nutzung klassischer Amazon SWF-Aktivitäten eignet.

04. März 2016

- Der AWS SDK for Java Abschnitt Amazon SWFBeispiele zur Verwendung des Themas wurde mit neuen Inhalten aktualisiert:
  - Amazon SWFGrundlagen — Enthält grundlegende Informationen darüber, wie Sie SWF in Ihre Projekte integrieren können.
  - Eine einfache Amazon SWF Anwendung erstellen — Ein neues Tutorial mit step-by-step Anleitungen für Java-Entwickler, die noch keine Erfahrung damit habenAmazon SWF.
  - Aktivitäts- und Workflow-Worker ordnungsgemäß herunterfahren — Beschreibt, wie Sie mithilfe der Parallelitätsklassen von Java Amazon SWF Worker-Klassen ordnungsgemäß herunterfahren können.

23. Februar 2016

- Die Quelle für das AWS SDK for Java Entwicklerhandbuch wurde verschoben nach. aws-java-developer-guide

28. Dezember 2015

- Die Einstellung der JVM-TTL für DNS-Namenssuchen wurde von „Erweitert“ in „Verwenden von“ verschoben und aus Gründen der AWS SDK for Java Übersichtlichkeit neu geschrieben.

  Verwenden des SDKs mit Apache Maven wurde mit Informationen über die Einbindung der SDK-Bill of Materials (BOM) in Ihr Projekt aktualisiert.

04. August 2015

- SSL-Zertifikatsanforderungen sind ein neues Thema im Abschnitt „Erste SchritteAWS“, in dem die Umstellung auf SHA256-signierte Zertifikate für SSL-Verbindungen beschrieben wird. Außerdem wird beschrieben, wie Java-Umgebungen mit Version 1.6 und früheren Versionen repariert werden können, sodass diese Zertifikate verwendet werden können, die für AWS den Zugriff nach dem 30. September 2015 erforderlich sind.

  > **Note**
  >
  > Java 1.7+ kommt bereits mit SHA256-signierten Zertifikaten zurecht.

14. Mai 2014

- Das Material zur Einführung und zu den ersten Schritten wurde stark überarbeitet, um die neue Struktur des Leitfadens zu unterstützen. Es enthält nun Anleitungen zur Einrichtung von AWS Zugangsdaten und zur Region für die Entwicklung.

  Die Besprechung der Codebeispiele wurde in ihr eigenes Thema im Abschnitt Zusätzliche Dokumentation und Ressourcen verschoben.

  Informationen zum Anzeigen des SDK-Revisionsverlaufs wurden in die Einführung verschoben.

9. Mai 2014

- Die Gesamtstruktur der AWS SDK for Java-Dokumentation wurde vereinfacht und die Themen Erste Schritte und Zusätzliche Dokumentation und Ressourcen wurden aktualisiert.

  Neue Themen wurden hinzugefügt:

  - Mit AWS Anmeldeinformationen arbeiten — beschreibt die verschiedenen Möglichkeiten, wie Sie Anmeldeinformationen für die angeben könnenAWS SDK for Java.

- Using IAM Roles to Grant Access to AWS Resources on Amazon EC2 — enthält Informationen zur sicheren Angabe von Anmeldeinformationen für Anwendungen, die auf EC2-Instances ausgeführt werden.

9. Sept. 2013

- In diesem Thema, dem Dokumentverlauf, werden die Änderungen am AWS SDK for Java Entwicklerhandbuch nachverfolgt. Es handelt sich um ein begleitendes Dokument zu release notes history (Verlauf der Versionshinweise).