Guida per gli sviluppatori per la versione 1.x

AWS SDK per Java 1. x

AWS SDK per Java 1. x: Guida per gli sviluppatori per la versione 1.x

Table of Contents

La AWS SDK per Java versione 1.x è entrata in modalità manutenzione il 31 luglio 2024 e sarà disponibile il 31 end-of-supportdicembre 2025. Ti consigliamo di eseguire la migrazione a per continuare AWS SDK for Java 2.xa ricevere nuove funzionalità, miglioramenti della disponibilità e aggiornamenti di sicurezza.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.

Guida per gli sviluppatori - AWS SDK per Java 1.x

AWS SDK per JavaFornisce un'API Java per i AWS servizi. Utilizzando l'SDK, puoi creare facilmente applicazioni Java che funzionano con Amazon S3 Amazon EC2 DynamoDB, e altro ancora. Regolarmente, aggiungiamo ad AWS SDK per Java il supporto per nuovi servizi. Per un elenco dei servizi supportati e delle relative versioni API incluse in ogni versione dell'SDK, consulta le note di rilascio relative alla versione con cui stai lavorando.

È stata rilasciata la versione 2 dell'SDK

Dai un'occhiata al nuovo AWS SDK per Java 2.x su https://github.com/aws/ aws-sdk-java-v 2/. Include funzionalità molto attese, come un modo per collegare un'implementazione HTTP. Per iniziare, consulta la Guida per sviluppatori AWS SDK per Java 2.x.

Documentazione e risorse aggiuntive

Oltre a questa guida, le seguenti sono preziose risorse online per AWS SDK per Java gli sviluppatori:

- AWS SDK per Java Documentazione di riferimento API
- Blog per gli sviluppatori Java
- Forum per gli sviluppatori Java
- GitHub:
  - Origine documentazione
  - Problemi relativi alla documentazione
  - Origine SDK
  - Problemi relativi all'SDK
  - Esempi SDK
  - Canale Gitter
- Il AWS Code Sample Catalog
- @awsforjava (Twitter)
- note di rilascio

Supporto per IDE Eclipse

Se sviluppate codice utilizzando l'IDE di Eclipse, potete utilizzarlo AWS Toolkit for Eclipseper aggiungerlo AWS SDK per Java a un progetto Eclipse esistente o per creare un nuovo AWS SDK per Java progetto. Il toolkit supporta anche la creazione e il caricamento di Lambda funzioni, l'avvio e il monitoraggio delle Amazon EC2 istanze, la gestione di IAM utenti e gruppi di sicurezza, un AWS CloudFormation editor di modelli e altro ancora.

Consulta la Guida per l'AWS Toolkit for Eclipse utente per la documentazione completa.

Sviluppo di applicazioni per Android

Se sei uno sviluppatore Android, Amazon Web Services pubblica un SDK creato appositamente per lo sviluppo Android: Amplify Android (Mobile SDK for Android).AWS

Visualizzazione della cronologia delle revisioni dell'SDK

Per visualizzare la cronologia delle versioni di SDK AWS SDK per Java, comprese le modifiche e i servizi supportati, consulta le note di rilascio dell'SDK.

Creazione della documentazione di riferimento in Java per le versioni precedenti dell'SDK

L'AWS SDK per Java API Reference rappresenta la build più recente della versione 1.x dell'SDK. Se utilizzi una build precedente della versione 1.x, potresti voler accedere alla documentazione di riferimento dell'SDK corrispondente alla versione che stai utilizzando.

Il modo più semplice per creare la documentazione è utilizzare lo strumento di compilazione Maven di Apache. Scarica e installa prima Maven se non lo hai già sul tuo sistema, quindi usa le seguenti istruzioni per creare la documentazione di riferimento.

1. Individua e seleziona la versione SDK che stai utilizzando nella pagina delle versioni del repository SDK su. GitHub
2. Scegli il link `zip` (per la maggior parte delle piattaforme, incluso Windows) o `tar.gz` (Linux, macOS o Unix) per scaricare l'SDK sul tuo computer.
3. Decomprimi l'archivio in una directory locale.

4. Nella riga di comando, accedi alla directory in cui hai decompresso l'archivio e digita quanto segue.

```
mvn javadoc:javadoc
```

5. Una volta completata la creazione, troverai la documentazione HTML generata nella `aws-java-sdk/target/site/apidocs/` directory.

Nozioni di base

In questa sezione vengono fornite informazioni su come installare, configurare e utilizzare AWS SDK per Java.

Argomenti

Configurazione di base con cui lavorare Servizi AWS

Panoramica

Per sviluppare correttamente applicazioni che consentano l'accesso Servizi AWS tramite AWS SDK per Java, sono necessarie le seguenti condizioni:

- È necessario essere in grado di AWS accedere al portale di accesso disponibile in AWS IAM Identity Center.
- Le autorizzazioni del ruolo IAM configurato per l'SDK devono consentire l'accesso a Servizi AWS ciò che l'applicazione richiede. Le autorizzazioni associate alla policy PowerUserAccess AWS gestita sono sufficienti per la maggior parte delle esigenze di sviluppo.
- Un ambiente di sviluppo con i seguenti elementi:
  - File di configurazione condivisi configurati nel modo seguente:
    - Il `config` file contiene un profilo predefinito che specifica un Regione AWS.
    - Il `credentials` file contiene credenziali temporanee come parte di un profilo predefinito.
  - Un'installazione adeguata di Java.
  - Uno strumento di automazione degli edifici come Maven o Gradle.
  - Un editor di testo per lavorare con il codice.
  - (Facoltativo, ma consigliato) Un IDE (ambiente di sviluppo integrato) come IntelliJ IDEA, Eclipse o. NetBeans

Quando usi un IDE, puoi anche integrarlo con Kit di strumenti AWS s per lavorare più facilmente. Servizi AWS I Kit di strumenti AWS per IntelliJe AWS Toolkit for Eclipsesono due toolkit che è possibile utilizzare per lo sviluppo in Java.

> **Important**
>
> Le istruzioni in questa sezione di configurazione presuppongono che l'utente o l'organizzazione utilizzi IAM Identity Center. Se la tua organizzazione utilizza un provider di identità esterno che funziona indipendentemente da IAM Identity Center, scopri come ottenere credenziali temporanee da utilizzare con l'SDK for Java. Segui queste istruzioni per aggiungere credenziali temporanee al file. `~/.aws/credentials`
> Se il tuo provider di identità aggiunge automaticamente credenziali temporanee al `~/.aws/credentials` file, assicurati che il nome del profilo sia `[default]` tale da non dover fornire un nome di profilo all'SDK o. AWS CLI

Capacità di accesso al portale di accesso AWS

Il portale di AWS accesso è il luogo web in cui è possibile accedere manualmente allo IAM Identity Center. Il formato dell'URL è `d-xxxxxxxxxx.awsapps.com/start` o*`your_subdomain`*`.awsapps.com/start`.

Se non conosci il portale di AWS accesso, segui le linee guida per l'accesso all'account nella fase 1 dell'argomento sull'autenticazione di IAM Identity Center nella guida di riferimento AWS SDKs e agli strumenti. Non seguite lo Step 2 perché la versione AWS SDK per Java 1.x non supporta l'aggiornamento automatico dei token e il recupero automatico delle credenziali temporanee per l'SDK descritto nella Fase 2.

Configura file di configurazione condivisi

I file di configurazione condivisi risiedono sulla workstation di sviluppo e contengono le impostazioni di base utilizzate da tutti AWS SDKs e dalla (AWS Command Line Interface CLI). I file di configurazione condivisi possono contenere diverse impostazioni, ma queste istruzioni configurano gli elementi di base necessari per lavorare con l'SDK.

Configura il file condiviso **`config`**

L'esempio seguente mostra il contenuto di un `config` file condiviso.

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

Per scopi di sviluppo, utilizzate il codice Regione AWS più vicino al punto in cui intendete eseguire il codice. Per un elenco dei codici regionali da utilizzare nel `config` file, consulta la Riferimenti generali di Amazon Web Services guida. L'`json`impostazione per il formato di output è uno dei diversi valori possibili.

Segui le indicazioni riportate in questa sezione per creare il `config` file.

Imposta le credenziali temporanee per l'SDK

Dopo aver avuto accesso a un Account AWS ruolo IAM tramite il portale di AWS accesso, configura il tuo ambiente di sviluppo con credenziali temporanee per l'accesso all'SDK.

Passaggi per configurare un **`credentials`** file locale con credenziali temporanee

1. Crea un `credentials` file condiviso.
2. Nel `credentials` file, incolla il seguente testo segnaposto finché non inserisci le credenziali temporanee di lavoro.

   ```
   [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. Salvare il file. Il file `~/.aws/credentials` dovrebbe ora esistere sul tuo sistema di sviluppo locale. Questo file contiene il profilo [predefinito] utilizzato dall'SDK for Java se non viene specificato un profilo denominato specifico.
4. Accedere al portale di AWS accesso.
5. Segui queste istruzioni sotto l'intestazione Aggiornamento manuale delle credenziali per copiare le credenziali del ruolo IAM dal AWS portale di accesso.

   a. Per la fase 4 delle istruzioni collegate, scegli il nome del ruolo IAM che concede l'accesso per le tue esigenze di sviluppo. Questo ruolo in genere ha un nome simile PowerUserAccessa Developer.

   b. Per il passaggio 7, seleziona l'opzione Aggiungi manualmente un profilo al file delle AWS credenziali e copia il contenuto.

6. Incolla le credenziali copiate nel `credentials` file locale e rimuovi qualsiasi nome di profilo che è stato incollato. Il file dovrebbe essere simile al seguente:

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

7. Salvare il file `credentials`

L'SDK for Java accederà a queste credenziali temporanee quando creerà un client di servizio e le utilizzerà per ogni richiesta. Le impostazioni per il ruolo IAM scelto nella fase 5a determinano la durata di validità delle credenziali temporanee. La durata massima è di dodici ore.

Dopo la scadenza delle credenziali temporanee, ripeti i passaggi da 4 a 7.

Installare un ambiente di sviluppo Java

La versione AWS SDK per Java V1 richiede un JDK Java 7 o versione successiva e sono supportate tutte le versioni JDK di Java LTS (supporto a lungo termine). Se si utilizza la versione 1.12.767 o precedente dell'SDK, è possibile utilizzare Java 7, ma se si utilizza la versione 1.12.768 o successiva dell'SDK, è necessario Java 8. L'archivio centrale di Maven elenca l'ultima versione dell'SDK for Java.

AWS SDK per Java Funziona con l'Oracle Java SE Development Kit e con le distribuzioni di Open Java Development Kit (OpenJDK) come Amazon Corretto, Red Hat OpenJDK e Adoptium.

Modi per ottenere il AWS SDK per Java

Prerequisiti

Per utilizzare il AWS SDK per Java, devi avere:

- Devi essere in grado di accedere al portale di AWS accesso disponibile in AWS IAM Identity Center.
- Un'installazione adeguata di Java.
- Credenziali temporanee configurate nel `credentials` file condiviso locale.

Consulta l'the section called “Configurazione di base”argomento per istruzioni su come configurare l'uso dell'SDK for Java.

Usa uno strumento di compilazione per gestire le dipendenze per l'SDK for Java (consigliato)

Ti consigliamo di utilizzare Apache Maven o Gradle con il tuo progetto per accedere alle dipendenze richieste dell'SDK for Java. Questa sezione descrive come utilizzare questi strumenti.

Scarica ed estrai l'SDK (scelta non consigliata)

Ti consigliamo di utilizzare uno strumento di compilazione per accedere all'SDK del tuo progetto. Puoi, tuttavia, scaricare un jar predefinito dell'ultima versione dell'SDK.

> **Note**
>
> Per informazioni su come scaricare e creare versioni precedenti dell'SDK, consulta Installazione delle versioni precedenti dell'SDK.

1. Scarica l'SDK da .zip. https://sdk-for-java.amazonwebservices.com/latest/ aws-java-sdk
2. Dopo aver scaricato l'SDK, estrai i contenuti in una directory locale.

L'SDK contiene le seguenti directory:

- `documentation`- contiene la documentazione dell'API (disponibile anche sul Web: AWS SDK per Java API Reference).
- `lib`- contiene i `.jar` file SDK.
- `samples`- contiene un codice di esempio funzionante che dimostra come utilizzare l'SDK.
- `third-party/lib`- contiene librerie di terze parti utilizzate dall'SDK, come Apache commons logging, AspectJ e il framework Spring.

Per utilizzare l'SDK, aggiungi il percorso completo `lib` e le `third-party` directory alle dipendenze nel tuo file di build e aggiungile al tuo java per eseguire il codice. `CLASSPATH`

Crea versioni precedenti dell'SDK dal codice sorgente (scelta non consigliata)

Solo la versione più recente dell'SDK completo viene fornita in forma predefinita come jar scaricabile. Tuttavia, puoi creare una versione precedente dell'SDK utilizzando Apache Maven (open source).

Maven scaricherà tutte le dipendenze necessarie, creerà e installerà l'SDK in un unico passaggio. Visita http://maven.apache.org/ per istruzioni di installazione e ulteriori informazioni.

1. Vai alla GitHub pagina dell'SDK all'indirizzo: AWS SDK per Java (GitHub).
2. Scegli il tag corrispondente al numero di versione dell'SDK che desideri. Ad esempio `1.6.10`.
3. Fai clic sul pulsante Scarica ZIP per scaricare la versione dell'SDK selezionata.
4. Decomprimi il file in una directory del tuo sistema di sviluppo. Su molti sistemi, è possibile utilizzare il gestore di file grafico per eseguire questa operazione o utilizzare l'`unzip`utilità in una finestra di terminale.
5. In una finestra di terminale, accedi alla directory in cui hai decompresso il codice sorgente dell'SDK.
6. Compila e installa l'SDK con il seguente comando (è richiesto Maven):

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

   Il file `.jar` risultante viene creato nella directory `target`.
7. (Facoltativo) Crea la documentazione di riferimento dell'API utilizzando il seguente comando:

   ```
   mvn javadoc:javadoc
   ```

   La documentazione è incorporata nella `target/site/apidocs/` directory.

Usa gli strumenti di costruzione

L'uso di strumenti di compilazione aiuta a gestire lo sviluppo di progetti Java. Sono disponibili diversi strumenti di compilazione, ma mostriamo come iniziare a usare due strumenti di compilazione popolari: Maven e Gradle. Questo argomento mostra come utilizzare questi strumenti di compilazione per gestire le dipendenze SDK for Java necessarie per i tuoi progetti.

Argomenti

- Utilizzare gli SDK con Apache Maven
- Utilizzare gli SDK con Gradle

Utilizzare gli SDK con Apache Maven

Puoi usare Apache Maven per configurare e creare AWS SDK per Java progetti o per creare l'SDK stesso.

> **Note**
>
> Per utilizzare le linee guida in questo argomento è necessario che sia installato Maven. Se non è già installato, visita http://maven.apache.org/ per scaricarlo e installarlo.

Crea un nuovo pacchetto Maven

Per creare un pacchetto Maven di base, apri una finestra di terminale (riga di comando) ed esegui:

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

Sostituisci org.example.basicapp con lo spazio dei nomi completo del pacchetto della tua applicazione e myapp con il nome del tuo progetto (questo diventerà il nome della directory del tuo progetto).

Per impostazione predefinita, crea un modello di progetto utilizzando l'archetipo quickstart, che è un buon punto di partenza per molti progetti. Sono disponibili altri archetipi; visita la pagina degli archetipi di Maven per un elenco degli archetipi inclusi. Può scegliere un determinato archetipo da utilizzare aggiungendo l'argomento `-DarchetypeArtifactId` al comando `archetype:generate`. Per esempio:

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

> **Note**
>
> Molte altre informazioni sulla creazione e la configurazione dei progetti sono disponibili nella Maven Getting Started Guide.

Configura l'SDK come dipendenza Maven

Per utilizzarlo AWS SDK per Java nel tuo progetto, dovrai dichiararlo come dipendenza nel file del tuo progetto. `pom.xml` A partire dalla versione 1.9.0, puoi importare singoli componenti o l'intero SDK.

Specificazione di singoli moduli SDK

Per selezionare singoli moduli SDK, utilizza la AWS SDK per Java distinta base (BOM) per Maven, che garantirà che i moduli specificati utilizzino la stessa versione dell'SDK e siano compatibili tra loro.

Per utilizzare il BOM, aggiungi una `<dependencyManagement>` sezione al `pom.xml` file dell'applicazione, aggiungendola `aws-java-sdk-bom` come dipendenza e specificando la versione dell'SDK che desideri utilizzare:

```
<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>
```

Per visualizzare l'ultima versione della distinta base disponibile su AWS SDK per Java Maven Central, visita: com.amazonaws/. https://mvnrepository.com/artifact/ aws-java-sdk-bom Puoi anche utilizzare questa pagina per vedere quali moduli (dipendenze) sono gestiti dal BOM che puoi includere nella sezione del file del tuo progetto. `<dependencies>` `pom.xml`

Ora puoi selezionare singoli moduli dall'SDK che usi nella tua applicazione. Poiché la versione SDK è già stata dichiarata nella distinta base, non è necessario specificare il numero di versione di ogni componente.

```
<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>
```

Puoi anche fare riferimento a per AWS Code Sample Catalog scoprire quali dipendenze usare per un determinato periodo. Servizio AWS Fare riferimento al file POM in un esempio di servizio specifico. Ad esempio, se sei interessato alle dipendenze per il servizio AWS S3, consulta l'esempio completo su. GitHub (Guardate il pom 3). under /java/example_code/s

Importazione di tutti i moduli SDK

Se desideri inserire l'intero SDK come dipendenza, non utilizzare il metodo BOM, ma dichiaralo semplicemente nel tuo modo: `pom.xml`

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

Creare il progetto

Una volta impostato il progetto, puoi crearlo usando il comando di Maven: `package`

```
mvn package
```

Questo creerà il tuo `0jar` file nella `target` directory.

Crea l'SDK con Maven

Puoi usare Apache Maven per creare l'SDK dal codice sorgente. Per fare ciò, scarica il codice SDK da, decomprimilo localmente GitHub, quindi esegui il seguente comando Maven:

```
mvn clean install
```

Utilizzare gli SDK con Gradle

Per gestire le dipendenze SDK per il tuo progetto Gradle, importa la BOM Maven per the nel file dell'applicazione. AWS SDK per Java `build.gradle`

> **Note**
>
> Negli esempi seguenti, sostituisci *`1.12.529`* nel file di build con una versione valida di. AWS SDK per Java Trova la versione più recente nel repository centrale di Maven.

Configurazione del progetto per Gradle 4.6 o versioni successive

A partire da Gradle 4.6, è possibile utilizzare la funzionalità di supporto POM migliorata di Gradle per importare file BOM (Bill of Materials) dichiarando una dipendenza da una BOM.

1. Se stai usando Gradle 5.0 o versioni successive, vai al passaggio 2. Altrimenti, abilita la funzione IMPROVED_POM_SUPPORT nel file. `settings.gradle`

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

2. Aggiungete il BOM alla sezione delle dipendenze del file dell'applicazione. `build.gradle`

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

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

3. Specifica i moduli SDK da utilizzare nella sezione dipendenze. Ad esempio, quanto segue include una dipendenza per 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 risolve automaticamente la versione corretta delle dipendenze SDK utilizzando le informazioni della distinta base.

Di seguito è riportato un esempio di `build.gradle` file completo che include una dipendenza per. Amazon S3

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

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
  mavenCentral()
}

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

> **Note**
>
> Nell'esempio precedente, sostituite la dipendenza per Amazon S3 con le dipendenze dei AWS servizi che utilizzerete nel progetto. I moduli (dipendenze) gestiti dalla AWS SDK per Java BOM sono elencati nell'archivio centrale di Maven.

Configurazione del progetto per le versioni di Gradle precedenti alla 4.6

Le versioni di Gradle precedenti alla 4.6 non dispongono del supporto BOM nativo. Per gestire AWS SDK per Java le dipendenze del tuo progetto, usa il plug-in di gestione delle dipendenze di Spring per Gradle per importare la BOM Maven per l'SDK.

1. Aggiungi il plug-in di gestione delle dipendenze al file dell'applicazione. `build.gradle`

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

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

2. Aggiungere la distinta base alla sezione dependencyManagement del file.

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

3. Specificate i moduli SDK che utilizzerete nella sezione delle dipendenze. Ad esempio, quanto riportato di seguito include una dipendenza per Amazon S3.

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

Gradle risolve automaticamente la versione corretta delle dipendenze SDK utilizzando le informazioni della distinta base.

Di seguito è riportato un esempio di `build.gradle` file completo che include una dipendenza per. Amazon S3

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

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
```

```
  mavenCentral()
}

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

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

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

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

> **Note**
>
> Nell'esempio precedente, sostituite la dipendenza per Amazon S3 con le dipendenze del AWS servizio che utilizzerete nel progetto. I moduli (dipendenze) gestiti dalla AWS SDK per Java BOM sono elencati nell'archivio centrale di Maven.

Per ulteriori informazioni sulla specificazione delle dipendenze dell'SDK utilizzando il BOM, consulta Uso dell'SDK con Apache Maven.

Imposta credenziali AWS temporanee e Regione AWS per lo sviluppo

Per connettersi a uno qualsiasi dei servizi supportati con AWS SDK per Java, è necessario fornire credenziali AWS temporanee. CLIs Utilizzano AWS SDKs le catene di provider per cercare

credenziali AWS temporanee in diversi punti, tra cui variabili di ambiente di sistema/utente e file di configurazione locali. AWS

Questo argomento fornisce informazioni di base sulla configurazione delle credenziali AWS temporanee per lo sviluppo di applicazioni locali tramite. AWS SDK per Java Se devi configurare le credenziali da utilizzare all'interno di un' EC2 istanza o se utilizzi l'IDE Eclipse per lo sviluppo, consulta invece i seguenti argomenti:

- Quando usi un' EC2 istanza, crea un ruolo IAM e poi consenti all' EC2 istanza di accedere a quel ruolo, come mostrato in Using IAM Roles to Grant Access to AWS Resources on. Amazon EC2
- Configura AWS le credenziali all'interno di Eclipse utilizzando. AWS Toolkit for Eclipse Per ulteriori informazioni, consulta Configurare AWS le credenziali nella Guida per AWS Toolkit for Eclipse l'utente.

Configurare le credenziali temporanee

È possibile configurare le credenziali temporanee per i AWS SDK per Java in diversi modi, ma ecco gli approcci consigliati:

- Impostate le credenziali temporanee nel file di profilo delle AWS credenziali sul sistema locale, che si trova in:
  - `~/.aws/credentials` su Linux, macOS o Unix
  - `C:\Users\USERNAME\.aws\credentials` in Windows

  Per istruzioni su come ottenere le the section called “Imposta le credenziali temporanee per l'SDK” credenziali temporanee, consulta questa guida.
- Imposta le variabili `AWS_ACCESS_KEY_IDAWS_SECRET_ACCESS_KEY`, e di `AWS_SESSION_TOKEN` ambiente.

  Per impostare queste variabili su Linux, macOS o Unix, utilizza :

```
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
```

  Per impostare queste variabili su Windows, utilizza :

```
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
```

- EC2 Ad esempio, specifica un ruolo IAM e quindi consenti all' EC2 istanza di accedere a quel ruolo. Consulta IAM Roles Amazon EC2 nella Guida Amazon EC2 utente per le istanze Linux per una discussione dettagliata su come funziona.

Dopo aver impostato le credenziali AWS temporanee utilizzando uno di questi metodi, queste verranno caricate automaticamente AWS SDK per Java dalla catena di fornitori di credenziali predefinita. Per ulteriori informazioni sull'utilizzo delle AWS credenziali nelle applicazioni Java, consultate Lavorare con le credenziali. AWS

Aggiornamento delle credenziali IMDS

AWS SDK per Java Supporta l'aggiornamento facoltativo delle credenziali IMDS in background ogni 1 minuto, indipendentemente dalla data di scadenza delle credenziali. Ciò consente di aggiornare le credenziali più frequentemente e riduce la possibilità che il mancato accesso all'IMDS influisca sulla disponibilità percepita. AWS

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

Imposta il Regione AWS

È necessario impostare un valore predefinito Regione AWS che verrà utilizzato per accedere ai AWS servizi con AWS SDK per Java. Per ottimizzare le prestazioni di rete, scegliere una regione

geograficamente vicina alla propria posizione (o ai clienti). Per un elenco delle regioni per ogni servizio, consulta Regioni ed endpoint nel Riferimento Amazon Web Services generale.

> **Note**
>
> Se non si seleziona una regione, per impostazione predefinita verrà utilizzato us-east-1.

Puoi utilizzare tecniche simili per impostare le credenziali per impostare la tua regione predefinita: AWS

- Imposta il Regione AWS file di AWS configurazione sul tuo sistema locale, che si trova in:
  - ~/.aws/config su Linux, macOS o Unix
  - C:\Users\USERNAME\ .aws\ config su Windows

  Questo file deve contenere righe nel seguente formato:

  +

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

  +

  Sostituisci il valore desiderato Regione AWS (ad esempio, «us-east-1") con your_aws_region.
- Imposta la variabile di ambiente `AWS_REGION`.

  Su Linux, macOS o Unix, usa:

  ```
  export AWS_REGION=your_aws_region
  ```

  In Windows, utilizza :

  ```
  set AWS_REGION=your_aws_region
  ```

  Dove your_aws_region è il nome desiderato. Regione AWS

Utilizzando il AWS SDK per Java

Questa sezione fornisce importanti informazioni generali sulla programmazione e si applicano a tutti i servizi AWS SDK per Java che è possibile utilizzare con l'SDK.

Per informazioni ed esempi di programmazione specifici del servizio (per Amazon EC2,, Amazon S3, ecc.) Amazon SWF, consulta AWS SDK per Java Esempi di codice.

Argomenti

Migliori pratiche per AWS lo sviluppo con AWS SDK per Java

Le seguenti best practice possono aiutarti a evitare problemi o problemi durante lo sviluppo di AWS applicazioni con. AWS SDK per Java Abbiamo organizzato le migliori pratiche per servizio.

S3

Evita ResetExceptions

Quando carichi oggetti utilizzando gli Amazon S3 stream (tramite un `AmazonS3` client `oTransferManager`), potresti riscontrare problemi di connettività di rete o di timeout. Per impostazione predefinita, i AWS SDK per Java tentativi di riprovare i trasferimenti non sono riusciti

contrassegnando il flusso di input prima dell'inizio di un trasferimento e quindi reimpostandolo prima di riprovare.

Se lo stream non supporta mark and reset, l'SDK genera un messaggio ResetExceptionquando si verificano errori temporanei e i nuovi tentativi sono abilitati.

Procedura ottimale

Ti consigliamo di utilizzare stream che supportano le operazioni di mark e reset.

Il modo più affidabile per evitare a ResetExceptionè fornire dati utilizzando un File or FileInputStream, che AWS SDK per Java possono gestire senza essere vincolati dai limiti di marcatura e ripristino.

Se lo stream non è un FileInputStreamformato ma supporta mark and reset, puoi impostare il limite dei mark utilizzando il `setReadLimit` metodo di. RequestClientOptions Il suo valore predefinito è 128 KB. L'impostazione del valore del limite di lettura su un byte maggiore della dimensione dello stream eviterà in modo affidabile un. ResetException

Ad esempio, se la dimensione massima prevista di uno stream è 100.000 byte, imposta il limite di lettura su 100.001 (100.000 + 1) byte. La marcatura e il ripristino funzioneranno sempre per 100.000 byte o meno. Tieni presente che ciò potrebbe far sì che alcuni stream inseriscano nel buffer quel numero di byte nella memoria.

Creazione di client del servizio

Per effettuare richieste a Amazon Web Services, devi prima creare un oggetto client di servizio. Il metodo consigliato è utilizzare il service client builder.

Ciascuno Servizio AWS ha un'interfaccia di servizio con metodi per ogni azione nell'API del servizio. Ad esempio, viene denominata l'interfaccia di servizio per DynamoDB. AmazonDynamoDBClient Ogni interfaccia di servizio dispone di un client builder corrispondente che è possibile utilizzare per creare un'implementazione dell'interfaccia di servizio. La classe client builder for DynamoDB è denominata Builder. AmazonDynamo DBClient

Ottenere un generatore client

Per ottenere un'istanza del client builder, utilizzate il metodo static factory`standard`, come illustrato nell'esempio seguente.

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

Una volta che hai un builder, puoi personalizzare le proprietà del client utilizzando molti setter fluenti nell'API del builder. Ad esempio, puoi impostare un'area personalizzata e un provider di credenziali personalizzate, come segue.

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

> **Note**
>
> I `withXXX` metodi fluent restituiscono l'`builder`oggetto in modo da poter concatenare le chiamate ai metodi per comodità e per rendere il codice più leggibile. Dopo aver configurato le proprietà desiderate, puoi chiamare il metodo `build` per creare il client. Una volta creato, un client è immutabile e qualsiasi chiamata a `setRegion` o avrà esito negativo. `setEndpoint`

Un builder può creare più client con la stessa configurazione. Quando scrivi la tua applicazione, tieni presente che il builder è mutabile e non thread-safe.

Il codice seguente utilizza il builder come factory per le istanze dei client.

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

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

Il builder espone anche i setter fluenti per ClientConfiguratione e un elenco personalizzato di RequestMetricCollector2. RequestHandler

Di seguito è riportato un esempio completo che sovrascrive tutte le proprietà configurabili.

```
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();
```

Creazione di client asincroni

AWS SDK per Java Dispone di client asincroni (o asincroni) per ogni servizio (tranne Amazon S3) e un generatore di client asincroni corrispondente per ogni servizio.

Per creare un client DynamoDB asincrono con il valore predefinito ExecutorService

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

Oltre alle opzioni di configurazione supportate dal generatore di client sincroni (o sincronizzati), il client asincrono consente di impostare una configurazione personalizzata ExecutorFactoryper modificare le impostazioni utilizzate dal client asincrono. `ExecutorService` `ExecutorFactory`è un'interfaccia funzionale, quindi interagisce con le espressioni lambda e i riferimenti ai metodi di Java 8.

Per creare un client asincrono con un executor personalizzato

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

Usando DefaultClient

Sia i costruttori di client di sincronizzazione che quelli asincroni hanno un altro metodo di fabbrica denominato. `defaultClient` Questo metodo crea un client di servizio con la configurazione predefinita, utilizzando la catena di provider predefinita per caricare le credenziali e il. Regione AWS Se non è possibile determinare le credenziali o la regione dall'ambiente di esecuzione dell'applicazione, la chiamata a `defaultClient` non riesce. Per ulteriori informazioni su come

vengono AWS determinate le credenziali e l'area geografica, vedere Working with Credentials and Regione AWS Selection.

Per creare un client di servizio predefinito

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

Ciclo di vita del client

I client di servizio nell'SDK sono thread-safe e, per ottenere prestazioni ottimali, dovresti trattarli come oggetti di lunga durata. Ogni client dispone della propria risorsa del pool di connessioni. Chiudi esplicitamente i client quando non sono più necessari per evitare perdite di risorse.

Per chiudere in modo esplicito un client, chiamate il metodo. `shutdown` Dopo la chiamata`shutdown`, tutte le risorse del client vengono rilasciate e il client è inutilizzabile.

Per chiudere un client

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

Fornire credenziali temporanee a AWS SDK per Java

Per effettuare richieste a Amazon Web Services, è necessario fornire credenziali AWS temporanee AWS SDK per Java da utilizzare quando chiama i servizi. Questa operazione può essere eseguita nei modi seguenti:

- Utilizzando la catena di provider delle credenziali predefinita (scelta consigliata).
- Utilizzando un provider di credenziali o catena di provider specifica (o creando la propria).
- Fornisci tu stesso le credenziali temporanee nel codice.

Utilizzo della catena di provider delle credenziali predefinita

Quando inizializzate un nuovo client di servizio senza fornire alcun argomento, AWS SDK per Java tenta di trovare credenziali temporanee utilizzando la catena di fornitori di credenziali predefinita implementata dalla classe Default. AWSCredentials ProviderChain La catena di provider delle credenziali predefinita cerca le credenziali in questo ordine:

1. Variabili di ambiente -`AWS_ACCESS_KEY_ID`, or, e`AWS_SECRET_KEY`. `AWS_SECRET_ACCESS_KEY` `AWS_SESSION_TOKEN` AWS SDK per Java utilizza la EnvironmentVariableCredentialsProviderclasse per caricare queste credenziali.
2. Proprietà del sistema Java -`aws.accessKeyId`, `aws.secretKey` (ma non`aws.secretAccessKey`), e`aws.sessionToken`. li AWS SDK per Java utilizza SystemPropertiesCredentialsProviderper caricare queste credenziali.
3. Credenziali Web Identity Token dall'ambiente o dal contenitore.
4. Il file di profili di credenziali predefinito, generalmente situato in `~/.aws/credentials` (la posizione può variare in base alla piattaforma) e condiviso da molti utenti AWS SDKs e da. AWS CLI AWS SDK per Java Utilizza il ProfileCredentialsProviderper caricare queste credenziali.

   È possibile creare un file di credenziali utilizzando il `aws configure` AWS CLI comando fornito da oppure modificarlo con un editor di testo. Per informazioni sul formato del file delle credenziali, consulta Formato del file AWS delle credenziali.
5. Credenziali del contenitore Amazon ECS: caricate da Amazon ECS se la variabile `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` di ambiente è impostata. Le AWS SDK per Java utilizza per caricare queste ContainerCredentialsProvidercredenziali. È possibile specificare l'indirizzo IP per questo valore.
6. Credenziali del profilo di istanza: utilizzate sulle EC2 istanze e fornite tramite il servizio di Amazon EC2 metadati. Le AWS SDK per Java utilizza per caricare queste InstanceProfileCredentialsProvidercredenziali. È possibile specificare l'indirizzo IP per questo valore.

> **Note**
>
> Le credenziali del profilo di istanza vengono utilizzate solo se non `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` sono impostate. Per ulteriori informazioni, consulta EC2ContainerCredentialsProviderWrapper.

Imposta credenziali temporanee

Per poter utilizzare le credenziali AWS temporanee, devono essere impostate in almeno una delle posizioni precedenti. Per informazioni sull'impostazione delle credenziali, consulta i seguenti argomenti:

- Per specificare le credenziali nell'ambiente o nel file dei profili di credenziali predefinito, vedere. the section called “Configurare le credenziali temporanee”
- Per impostare proprietà del sistema Java, consulta il tutorial sulle proprietà del sistema nel sito Web dei tutorial Java ufficiale.
- Per configurare e utilizzare le credenziali del profilo di istanza con le tue EC2 istanze, consulta Using IAM Roles to Grant Access to Resources on. AWS Amazon EC2

Imposta un profilo di credenziali alternativo

AWS SDK per Java Utilizza il profilo predefinito per impostazione predefinita, ma esistono modi per personalizzare il profilo proveniente dal file delle credenziali.

È possibile utilizzare la variabile AWS di ambiente Profile per modificare il profilo caricato dall'SDK.

Ad esempio, su Linux, macOS o Unix è necessario eseguire il comando seguente per modificare il profilo in MyProfile.

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

In Windows si utilizzerebbe quanto segue.

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

L'impostazione della variabile di `AWS_PROFILE` ambiente influisce sul caricamento delle credenziali per tutti gli strumenti AWS SDKs e strumenti ufficialmente supportati (inclusi i AWS CLI e i AWS Tools for Windows PowerShell). Per modificare solo il profilo di un'applicazione Java, potete `aws.profile` invece utilizzare la proprietà di sistema.

> **Note**
>
> La variabile di ambiente prevale sulla proprietà di sistema.

Imposta una posizione alternativa per il file delle credenziali

AWS SDK per Java Carica automaticamente le credenziali AWS temporanee dalla posizione predefinita del file delle credenziali. Tuttavia, puoi anche specificare il percorso impostando la

variabile di ambiente `AWS_CREDENTIAL_PROFILES_FILE` con il percorso completo al file delle credenziali.

È possibile utilizzare questa funzionalità per modificare temporaneamente la posizione in cui AWS SDK per Java cerca il file delle credenziali (ad esempio, impostando questa variabile con la riga di comando). In alternativa, puoi impostare la variabile di ambiente nell'ambiente utente o di sistema per modificarla a livello di utente o di sistema.

Per ignorare il percorso del file delle credenziali predefinito

- Imposta la variabile di `AWS_CREDENTIAL_PROFILES_FILE` ambiente sulla posizione del file delle AWS credenziali.
  - Su Linux, macOS o Unix, usa:

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

  - In Windows, usa:

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

`Credentials`formato di file

Seguendo le istruzioni riportate nella configurazione di base di questa guida, il file delle credenziali dovrebbe avere il seguente formato di base.

```
[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>
```

Il nome del profilo è specificato tra parentesi quadre (ad esempio, `[default]`), seguito dai campi configurabili nel profilo come coppie chiave-valore. Nel `credentials` file possono essere presenti più profili, che possono essere aggiunti o modificati selezionando `aws configure --profile` *`PROFILE_NAME`* il profilo da configurare.

È possibile specificare campi aggiuntivi`metadata_service_timeout`, ad esempio `emetadata_service_num_attempts`. Questi non sono configurabili con la CLI: è necessario modificare il file manualmente se si desidera utilizzarli. Per ulteriori informazioni sul file di configurazione e sui campi disponibili, vedere Configurazione di AWS Command Line Interface nella Guida per l'utente. AWS Command Line Interface

Caricare le credenziali

Dopo aver impostato le credenziali temporanee, l'SDK le carica utilizzando la catena di provider di credenziali predefinita.

A tale scopo, create un'istanza di un Servizio AWS client senza fornire esplicitamente le credenziali al builder, come segue.

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

Specificate un fornitore di credenziali o una catena di fornitori

Puoi specificare un provider delle credenziali diverso dalla catena di provider delle credenziali predefinita utilizzando il generatore client.

Fornisci un'istanza di un provider di credenziali o di una catena di provider a un client builder che accetta un'interfaccia AWSCredentialsProvider come input. Nell'esempio seguente viene mostrato come utilizzare specificatamente le credenziali ambiente.

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

Per l'elenco completo dei provider di credenziali e delle catene di provider AWS SDK per Java forniti, consulta All Known Implementation Classes in Provider. AWSCredentials

> **Note**
>
> È possibile utilizzare questa tecnica per fornire provider di credenziali o catene di provider create utilizzando il proprio provider di credenziali che

implementa l'`AWSCredentialsProvider`interfaccia o sottoclassando la classe. AWSCredentialsProviderChain

Specificate esplicitamente le credenziali temporanee

Se la catena di credenziali predefinita o un provider o una catena di fornitori specifici o personalizzati non funzionano per il codice, puoi impostare le credenziali fornite in modo esplicito. Se hai recuperato credenziali temporanee utilizzando AWS STS, usa questo metodo per specificare le credenziali di accesso. AWS

1. Crea un'istanza della BasicSessionCredentialsclasse e forniscile la chiave di AWS accesso, la chiave AWS segreta e il token di AWS sessione che l'SDK utilizzerà per la connessione.
2. Crea un AWSStaticCredentialsProvidercon l'oggetto. `AWSCredentials`
3. Configurare il generatore client con `AWSStaticCredentialsProvider` e creare il client.

Di seguito è riportato un esempio.

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

Ulteriori informazioni

- Iscriviti AWS e crea un utente IAM
- Configura AWS le credenziali e la regione per lo sviluppo
- Utilizzo dei ruoli IAM per concedere l'accesso alle AWS risorse su Amazon EC2

Regione AWS Selezione

Le regioni consentono di accedere ai AWS servizi che risiedono fisicamente in un'area geografica specifica. Ciò può essere utile per la ridondanza e per mantenere i dati e le applicazioni in esecuzione vicino ai punti di accesso ai servizi stessi.

Verifica della disponibilità del servizio in una regione

Per verificare se un determinato prodotto Servizio AWS è disponibile in una regione, utilizza il `isServiceSupported` metodo relativo alla regione che desideri utilizzare.

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

Consultate la documentazione della classe Regions per le regioni che potete specificare e utilizzate il prefisso endpoint del servizio per eseguire le query. Il prefisso dell'endpoint di ogni servizio è definito nell'interfaccia del servizio. Ad esempio, il prefisso dell' DynamoDB endpoint è definito nel DB. AmazonDynamo

Scelta di una regione

A partire dalla versione 1.4 di AWS SDK per Java, puoi specificare un nome di regione e l'SDK sceglierà automaticamente l'endpoint appropriato per te. Per scegliere tu stesso l'endpoint, vedi Scelta di un endpoint specifico.

Per impostare in modo esplicito una regione, ti consigliamo di utilizzare l'enumerazione Regioni. Si tratta di un'enumerazione di tutte le regioni disponibili pubblicamente. Per creare un client con una regione dall'enum, usa il codice seguente.

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

Se la regione che stai tentando di utilizzare non è nell'`Regions`enum, puoi impostare la regione usando una stringa che rappresenta il nome della regione.

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

> **Note**
>
> Dopo che è stato creato con il generatore, il client è immutabile e la regione non può essere modificata. Se state lavorando con più client Regioni AWS per lo stesso servizio, dovreste creare più client, uno per regione.

Scelta di un endpoint specifico

Ogni AWS client può essere configurato per utilizzare un endpoint specifico all'interno di una regione chiamando il `withEndpointConfiguration` metodo durante la creazione del client.

Ad esempio, per configurare il Amazon S3 client per l'utilizzo della regione Europa (Irlanda), utilizzare il codice seguente.

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

Vedi Regioni ed endpoint per l'elenco corrente delle regioni e gli endpoint corrispondenti per tutti i AWS servizi.

Determina automaticamente la regione dall'ambiente

> **Important**
>
> Questa sezione si applica solo quando si utilizza un client builder per accedere ai AWS servizi. AWS i client creati utilizzando il costruttore del client non determineranno automaticamente la regione dall'ambiente e utilizzeranno invece la regione SDK predefinita (). USEast1

Quando è in esecuzione su Amazon EC2 o Lambda, potresti voler configurare i client in modo che utilizzino la stessa regione su cui è in esecuzione il codice. Questo consente di separare il codice dall'ambiente in cui è in esecuzione e facilita la distribuzione dell'applicazione in più regioni per minore latenza o ridondanza.

È necessario utilizzare i client builder per fare in modo che l'SDK rilevi automaticamente la regione in cui è in esecuzione il codice.

Per utilizzare la catena di credential/region provider predefinita per determinare la regione dall'ambiente, utilizzate il metodo del client builder. `defaultClient`

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

È lo stesso che si usa `standard` seguito da`build`.

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

Se non impostate esplicitamente una regione utilizzando `withRegion` i metodi, l'SDK consulta la catena di fornitori di regioni predefinita per cercare di determinare la regione da utilizzare.

Catena di provider delle regioni predefinita

Di seguito è riportato il processo di ricerca della regione:

1. Qualsiasi regione esplicita impostata utilizzando `withRegion` o `setRegion` sul builder stesso ha la precedenza su qualsiasi altra cosa.
2. La variabile di ambiente `AWS_REGION` è selezionata. Se è impostata, questa regione viene utilizzata per configurare il client.

   > **Note**
   >
   > Questa variabile di ambiente è impostata dal contenitore. Lambda

3. L'SDK controlla il file di configurazione AWS condiviso (che di solito si trova in`~/.aws/config`). Se la proprietà region è presente, viene utilizzata dall'SDK.
   - La variabile di ambiente `AWS_CONFIG_FILE` può essere utilizzata per personalizzare il percorso del file di configurazione condiviso.
   - La variabile di `AWS_PROFILE` ambiente o la proprietà di `aws.profile` sistema possono essere utilizzate per personalizzare il profilo caricato dall'SDK.
4. L'SDK tenta di utilizzare il servizio di metadati dell' Amazon EC2 istanza per determinare la regione dell'istanza attualmente in esecuzione. Amazon EC2
5. Se a questo punto l'SDK non ha ancora trovato una regione, la creazione del client non riesce con un'eccezione.

Durante lo sviluppo di AWS applicazioni, un approccio comune consiste nell'utilizzare il file di configurazione condiviso (descritto in Utilizzo della catena di provider di credenziali predefinita)

per impostare la regione per lo sviluppo locale e fare affidamento sulla catena di provider di aree predefinita per determinare la regione durante l'esecuzione sull'infrastruttura. AWS Questo semplifica notevolmente la creazione del client e mantiene l'applicazione portatile.

Gestione delle eccezioni

Comprendere come e quando vengono AWS SDK per Java generate le eccezioni è importante per creare applicazioni di alta qualità utilizzando l'SDK. Nelle seguenti sezioni vengono descritti i diversi casi di eccezioni che vengono generate dall'SDK e come gestirle in modo appropriato.

Perché eccezioni non controllate?

AWS SDK per Java Utilizza eccezioni di runtime (o non controllate) anziché eccezioni verificate per i seguenti motivi:

- Per consentire agli sviluppatori di controllare ogni dettaglio degli errori che desiderano gestire senza costringerli a gestire casi eccezionali che non destino preoccupazione (rendendo il codice eccessivamente dettagliato)
- Per prevenire problemi di scalabilità intrinseca con eccezioni controllate in applicazioni di grandi dimensioni

In generale, le eccezioni controllate funzionano bene su scale ridotte, ma possono diventare problematiche all'aumentare delle dimensioni e della complessità delle applicazioni.

Per ulteriori informazioni sull'uso delle eccezioni selezionate e deselezionate, consulta:

- Eccezioni non controllate: la controversia
- Il problema delle eccezioni controllate
- Le eccezioni verificate di Java erano un errore (ed ecco cosa vorrei fare al riguardo)

AmazonServiceException (e sottoclassi)

AmazonServiceExceptionè l'eccezione più comune che si verificherà quando si utilizza. AWS SDK per Java Questa eccezione rappresenta una risposta di errore da parte di un Servizio AWS. Ad esempio, se si tenta di terminare un' Amazon EC2 istanza che non esiste, EC2 verrà restituita una risposta di errore e tutti i dettagli di tale risposta di errore verranno inclusi nella `AmazonServiceException` risposta generata. Per alcuni casi, viene generata una sottoclasse di

`AmazonServiceException` per consentire agli sviluppatori di controllare tutti i dettagli di gestione dei casi di errore tramite blocchi catch.

Quando incontri un`AmazonServiceException`, sai che la tua richiesta è stata inviata con successo a Servizio AWS ma non può essere elaborata correttamente. Questo può essere dovuto a errori nei parametri della richiesta o a errori sul lato servizio.

`AmazonServiceException` fornisce informazioni quali:

- Codice di stato HTTP restituito
- Codice AWS di errore restituito
- Messaggio di errore dettagliato dal servizio
- AWS ID della richiesta non riuscita

`AmazonServiceException`include anche informazioni sul fatto che la richiesta non riuscita sia stata colpa del chiamante (una richiesta con valori non validi) o colpa Servizio AWS del chiamante (un errore interno del servizio).

AmazonClientException

AmazonClientExceptionindica che si è verificato un problema all'interno del codice client Java, durante il tentativo di inviare una richiesta a AWS o durante il tentativo di analizzare una risposta da. AWS Un `AmazonClientException` è generalmente più grave di un `AmazonServiceException` e indica un problema importante che impedisce al client di effettuare chiamate di servizio ai AWS servizi. Ad esempio, AWS SDK per Java genera una connessione di rete `AmazonClientException` se non è disponibile alcuna connessione di rete quando si tenta di richiamare un'operazione su uno dei client.

Programmazione asincrona

È possibile utilizzare metodi sincroni o asincroni per richiamare le operazioni sui servizi. AWS I metodi sincroni bloccano l'esecuzione del thread finché il client non riceve una risposta dal servizio. I metodi asincroni terminano immediatamente, riassegnando il controllo al thread chiamante senza attendere una risposta.

Poiché un metodo asincrono termina prima che sia disponibile una risposta, occorre un modo per ottenere la risposta quando è pronta. AWS SDK per Java Fornisce due modi: oggetti futuri e metodi di callback.

Java Futures

I metodi asincroni AWS SDK per Java restituiscono un oggetto Future che contiene i risultati dell'operazione asincrona in futuro.

Chiamate il `Future` `isDone()` metodo per vedere se il servizio ha già fornito un oggetto di risposta. Quando la risposta è pronta, è possibile ottenere l'oggetto di risposta chiamando il `Future` `get()` metodo. È possibile utilizzare questo meccanismo per verificare periodicamente i risultati dell'operazione asincrona mentre l'applicazione continua a lavorare su altre cose.

Ecco un esempio di operazione asincrona che chiama una Lambda funzione, ricevendo un oggetto che può contenere un oggetto. `Future` InvokeResult L'`InvokeResult`oggetto viene recuperato solo dopo is. `isDone()` `true`

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

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

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

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

        System.out.print("Waiting for future");
        while (future_res.isDone() == false) {
            System.out.print(".");
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {
```

```
                System.err.println("\nThread.sleep() was interrupted!");
                System.exit(1);
            }
        }

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

        System.exit(0);
    }
}
```

Chiamate asincrone

Oltre a utilizzare l'`Future`oggetto Java per monitorare lo stato delle richieste asincrone, l'SDK consente anche di implementare una classe che utilizza l'interfaccia. AsyncHandler `AsyncHandler`fornisce due metodi che vengono chiamati a seconda del modo in cui la richiesta è stata completata: e. `onSuccess` `onError`

Il vantaggio principale dell'approccio dell'interfaccia di callback è che vi evita di dover interrogare l'`Future`oggetto per scoprire quando la richiesta è stata completata. Al contrario, il codice può iniziare immediatamente la sua attività successiva e fare affidamento sull'SDK per chiamare il gestore al momento giusto.

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

Best practice

Esecuzione del callback

L'implementazione di `AsyncHandler` viene eseguita all'interno del pool di thread di proprietà del client asincrono. Il codice breve ed eseguito rapidamente è il più appropriato all'interno dell'implementazione. `AsyncHandler` Il codice a esecuzione prolungata o bloccante all'interno dei metodi di gestione può causare conflitti per il pool di thread utilizzato dal client asincrono e impedire al client di eseguire le richieste. Se hai un'attività di lunga durata che deve iniziare da un callback, chiedi al callback di eseguire la sua attività in un nuovo thread o in un pool di thread gestito dall'applicazione.

Configurazione del pool di thread

I client asincroni inclusi in AWS SDK per Java forniscono un pool di thread predefinito che dovrebbe funzionare per la maggior parte delle applicazioni. È possibile implementare un file personalizzato ExecutorServicee passarlo a client AWS SDK per Java asincroni per un maggiore controllo sulla gestione dei pool di thread.

Ad esempio, è possibile fornire un'`ExecutorService`implementazione che utilizzi un'impostazione personalizzata ThreadFactoryper controllare il modo in cui vengono denominati i thread nel pool o per registrare informazioni aggiuntive sull'utilizzo dei thread.

Accesso asincrono

La TransferManagerclasse dell'SDK offre supporto asincrono con cui lavorare. Amazon S3`TransferManager`gestisce caricamenti e download asincroni, fornisce report dettagliati sullo stato di avanzamento dei trasferimenti e supporta i callback in diversi eventi.

AWS SDK per Java Registrazione delle chiamate

AWS SDK per Java È dotato di strumentazione con Apache Commons Logging, che è un livello di astrazione che consente l'uso di uno qualsiasi dei numerosi sistemi di registrazione in fase di esecuzione.

I sistemi di registrazione supportati includono, tra gli altri, Java Logging Framework e Apache Log4j. In questo argomento viene mostrato come utilizzare Log4j. Puoi utilizzare la funzionalità di registrazione dell'SDK senza apportare modifiche al codice dell'applicazione.

Per ulteriori informazioni su Log4j, visita il sito Web Apache.

> **Note**
>
> Questo argomento si concentra su Log4j 1.x. Log4j2 non supporta direttamente Apache Commons Logging, ma fornisce un adattatore che indirizza automaticamente la registrazione delle chiamate a Log4j2 utilizzando l'interfaccia Apache Commons Logging. Per ulteriori informazioni, consulta Commons Logging Bridge nella documentazione di Log4j2.

Scarica il file JAR Log4J

Per utilizzare Log4j con l'SDK, devi scaricare il JAR Log4j dal sito Web di Apache. L'SDK non include il JAR. Copia il file JAR in una posizione sul tuo classpath.

Log4j utilizza un file di configurazione, log4j.properties. File di configurazione di esempio sono mostrati di seguito. Copia questo file di configurazione in una directory sul tuo classpath. Il file JAR Log4j e il file log4j.properties non devono necessariamente trovarsi nella stessa directory.

Il file di configurazione log4j.properties specifica proprietà come il livello di registrazione, dove viene inviato l'output di registrazione (ad esempio, a un file o alla console) e il formato dell'output. Il livello di registrazione è la granularità di output generata dal logger. Log4j supporta il concetto di gerarchie di registrazione multiple. Il livello di registrazione è impostato in modo indipendente per ogni gerarchia. Le due gerarchie di registrazione seguenti sono disponibili in AWS SDK per Java:

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

Impostazione di classpath

Sia il file JAR Log4j che il file log4j.properties devono trovarsi nel classpath. Se stai usando Apache Ant, imposta il classpath nell'elemento del tuo file Ant. `path` L'esempio seguente mostra un elemento del percorso del file Ant per l' Amazon S3 esempio incluso nell'SDK.

```
<path id="aws.java.sdk.classpath">
```

```
  <fileset dir="../../third-party" includes="**/*.jar"/>
  <fileset dir="../../lib" includes="**/*.jar"/>
  <pathelement location="."/>
</path>
```

Se stai utilizzando l'IDE Eclipse, puoi impostare il classpath aprendo il menu e passando a Project (Progetto) | Properties (Proprietà) | Java Build Path (Percorso build Java).

Errori e avvertenze specifici del servizio

Ti consigliamo di lasciare sempre la gerarchia dei logger «com.amazonaws» impostata su «WARN» per catturare eventuali messaggi importanti dalle librerie dei client. Ad esempio, se il Amazon S3 client rileva che l'applicazione non ha chiuso correttamente un'applicazione `InputStream` e che potrebbe essere in corso una perdita di risorse, il client S3 lo segnala tramite un messaggio di avviso ai log. Questo garantisce inoltre che i messaggi vengono registrati se il client presenta problemi di gestione delle richieste o delle risposte.

Il seguente file log4j.properties lo imposta su WARN, che include `rootLogger` i messaggi di avviso e di errore provenienti da tutti i logger nella gerarchia «com.amazonaws». In alternativa, puoi impostare esplicitamente il logger com.amazonaws su WARN.

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

Registrazione del riepilogo di richieste/risposte

Ogni richiesta a un Servizio AWS genera un ID di AWS richiesta univoco, utile in caso di problemi relativi alla gestione di una richiesta. Servizio AWS AWS IDs le richieste sono accessibili a livello di codice tramite gli oggetti Exception nell'SDK per qualsiasi chiamata di servizio fallita e possono anche essere segnalate tramite il livello di registro DEBUG nel logger «com.amazonaws.request».

Il seguente file log4j.properties consente un riepilogo delle richieste e delle risposte, inclusa la richiesta. AWS IDs

```
log4j.rootLogger=WARN, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
```

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

Di seguito è riportato un esempio di output del log.

```
2009-12-17 09:53:04,269 [main] DEBUG com.amazonaws.request - Sending
Request: POST https://rds.amazonaws.com / Parameters: (MaxRecords: 20,
Action: DescribeEngineDefaultParameters, SignatureMethod: HmacSHA256,
AWSAccessKeyId: ACCESSKEYID, Version: 2009-10-16, SignatureVersion: 2,
Engine: mysql5.1, Timestamp: 2009-12-17T17:53:04.267Z, Signature:
q963XH63Lcovl5Rr71APlzlye99rmWwT9DfuQaNznkD, ) 2009-12-17 09:53:04,464
[main] DEBUG com.amazonaws.request - Received successful response: 200, {AWS}
Request ID: 694d1242-cee0-c85e-f31f-5dab1ea18bc6 2009-12-17 09:53:04,469
[main] DEBUG com.amazonaws.request - Sending Request: POST
https://rds.amazonaws.com / Parameters: (ResetAllParameters: true, Action:
ResetDBParameterGroup, SignatureMethod: HmacSHA256, DBParameterGroupName:
java-integ-test-param-group-0000000000000, AWSAccessKeyId: ACCESSKEYID,
Version: 2009-10-16, SignatureVersion: 2, Timestamp:
2009-12-17T17:53:04.467Z, Signature:
9WcgfPwTobvLVcpyhbrdN7P7l3uH0oviYQ4yZ+TQjsQ=, )

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

Registrazione in rete Verbose

In alcuni casi, può essere utile visualizzare le richieste e le risposte esatte inviate e ricevute AWS SDK per Java . Non è consigliabile abilitare questa registrazione nei sistemi di produzione perché la scrittura di richieste di grandi dimensioni (ad esempio, un file in fase di caricamento Amazon S3) o risposte può rallentare notevolmente un'applicazione. Se hai davvero bisogno di accedere a queste informazioni, puoi abilitarle temporaneamente tramite il logger Apache HttpClient 4. Abilitando il livello DEBUG sul logger `org.apache.http.wire` si abilita la registrazione di tutti i dati di richiesta e di risposta.

Il seguente file log4j.properties attiva la registrazione cablata completa in Apache HttpClient 4 e dovrebbe essere attivato solo temporaneamente perché può avere un impatto significativo sulle prestazioni dell'applicazione.

```
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
```

Registrazione delle metriche di latenza

Se state cercando di risolvere i problemi e volete vedere metriche come il processo che richiede più tempo o se il lato server o client ha la latenza maggiore, il registratore di latenza può essere utile. Imposta il `com.amazonaws.latency` logger su DEBUG per abilitare questo logger.

> **Note**
>
> Questo logger è disponibile solo se le metriche SDK sono abilitate. Per ulteriori informazioni sul pacchetto di metriche SDK, consulta Enabling Metrics for. AWS SDK per 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
```

Di seguito è riportato un esempio di output del log.

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

Configurazione del client

AWS SDK per Java Consente di modificare la configurazione predefinita del client, utile quando si desidera:

- Connect a Internet tramite proxy
- Modifica le impostazioni di trasporto HTTP, ad esempio il timeout della connessione e i nuovi tentativi di richiesta
- Specificare i suggerimenti sulla dimensione del buffer del socket TCP

Configurazione proxy

Quando si costruisce un oggetto client, è possibile passare un ClientConfigurationoggetto opzionale per personalizzare la configurazione del client.

Se ti connetti a Internet tramite un server proxy, dovrai configurare le impostazioni del server proxy (host proxy, porta e nome utente/password) tramite l'oggetto. `ClientConfiguration`

Configurazione del trasporto HTTP

È possibile configurare diverse opzioni di trasporto HTTP utilizzando l'ClientConfigurationoggetto. Di tanto in tanto vengono aggiunte nuove opzioni; per visualizzare l'elenco completo delle opzioni che è possibile recuperare o impostare, consulta l' AWS SDK per Java API Reference.

> **Note**
>
> Ciascuno dei valori configurabili ha un valore predefinito definito da una costante. Per un elenco dei valori costanti per`ClientConfiguration`, consulta Constant Field Values nel riferimento AWS SDK per Java API.

Numero massimo connessioni

È possibile impostare il numero massimo consentito di connessioni HTTP aperte utilizzando ClientConfiguration. setMaxConnectionsmetodo.

> **Important**
>
> Imposta il numero massimo di connessioni per il numero di transazioni simultanee per evitare problemi e performance scarse. Per il valore massimo di connessioni predefinito, consulta Constant Field Values nel riferimento AWS SDK per Java API.

Timeout e gestione degli errori

È possibile impostare opzioni relative ai timeout e alla gestione degli errori con le connessioni HTTP.

- Timeout di connessione

  Il timeout della connessione è la quantità di tempo (in millisecondi) che la connessione HTTP aspetterà per stabilire una connessione prima di rinunciare. L'impostazione predefinita è 10.000 ms.

  Per impostare tu stesso questo valore, usa il ClientConfiguration. setConnectionTimeoutmetodo.

- Connection Time to Live (TTL)

  Per impostazione predefinita, l'SDK tenterà di riutilizzare le connessioni HTTP il più a lungo possibile. In situazioni di errore in cui viene stabilita una connessione a un server che è stato messo fuori servizio, disporre di un TTL limitato può facilitare il ripristino dell'applicazione. Ad esempio, impostando un TTL di 15 minuti, anche se è stata stabilita una connessione a un server con problemi, sarà possibile ristabilire la connessione a un nuovo server entro 15 minuti.

  Per impostare il TTL della connessione HTTP, utilizzate il metodo .setConnectionTTL. ClientConfiguration

- Numero massimo di tentativi di errore

  Il numero massimo di tentativi predefinito per gli errori recuperabili è 3. È possibile impostare un valore diverso utilizzando il. ClientConfiguration setMaxErrorMetodo Retry.

Indirizzo locale

Per impostare l'indirizzo locale a cui il client HTTP si collegherà, usaClientConfiguration. setLocalAddress.

Suggerimenti sulla dimensione del buffer del socket TCP

Gli utenti esperti che desiderano ottimizzare i parametri TCP di basso livello possono inoltre impostare suggerimenti sulla dimensione del buffer TCP tramite l'oggetto. ClientConfiguration La maggior parte degli utenti non avrà mai bisogno di modificare questi valori, ma sono disponibili per utenti esperti.

Le dimensioni ottimali del buffer TCP per un'applicazione dipendono in larga misura dalla configurazione e dalle funzionalità della rete e del sistema operativo. Ad esempio, la maggior parte dei sistemi operativi moderni fornisce una logica di regolazione automatica per le dimensioni del buffer TCP. Ciò può avere un grande impatto sulle prestazioni delle connessioni TCP che vengono mantenute aperte abbastanza a lungo da consentire l'ottimizzazione automatica per ottimizzare le dimensioni del buffer.

Le grandi dimensioni del buffer (ad esempio, 2 MB) consentono al sistema operativo di bufferizzare più dati in memoria senza richiedere al server remoto di confermare la ricezione di tali informazioni, e quindi possono essere particolarmente utili quando la rete ha un'elevata latenza.

Questo è solo un suggerimento e il sistema operativo potrebbe non rispettarlo. Quando si utilizza questa opzione, gli utenti devono sempre verificare i limiti e le impostazioni predefinite configurati del sistema operativo. La maggior parte dei sistemi operativi ha un limite massimo di dimensione del buffer TCP configurato e non consente di superare tale limite a meno che non aumenti esplicitamente il limite massimo di dimensione del buffer TCP.

Sono disponibili molte risorse per facilitare la configurazione delle dimensioni del buffer TCP e delle impostazioni TCP specifiche del sistema operativo, tra cui:

- Ottimizzazione dell'host

Policy di controllo degli accessi

AWS le politiche di controllo degli accessi consentono di specificare controlli di accesso dettagliati sulle risorse. AWS Una politica di controllo degli accessi consiste in una raccolta di dichiarazioni, che assumono la forma:

L'account A è autorizzato a eseguire l'azione B sulla risorsa C laddove si applica la condizione D.

Dove:

- A è il principale, Account AWS ovvero la richiesta di accesso o di modifica di una delle AWS risorse dell'utente.
- B è l'azione: il modo in cui si accede o si modifica la AWS risorsa, ad esempio l'invio di un messaggio a una Amazon SQS coda o la memorizzazione di un oggetto in un Amazon S3 bucket.
- C è la risorsa: l' AWS entità a cui il principale desidera accedere, ad esempio una Amazon SQS coda o un oggetto in cui è memorizzato. Amazon S3
- D è un insieme di condizioni: i vincoli opzionali che specificano quando consentire o negare l'accesso al principale per accedere alla risorsa. Sono disponibili molte condizioni espressive, alcune specifiche per ogni servizio. Ad esempio, è possibile utilizzare le condizioni relative alla data per consentire l'accesso alle risorse solo dopo o prima di un orario specifico.

Amazon S3 Esempio

L'esempio seguente dimostra una politica che consente a chiunque di accedere alla lettura di tutti gli oggetti in un bucket, ma limita l'accesso al caricamento di oggetti in quel bucket a due Account AWS s specifici (oltre all'account del proprietario del bucket).

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

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

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

Amazon SQS Esempio

Un uso comune delle policy consiste nell'autorizzare una Amazon SQS coda a ricevere messaggi da un argomento di Amazon SNS.

```
Policy policy = new Policy().withStatements(
    new Statement(Effect.Allow)
```

```
        .withPrincipals(Principal.AllUsers)
        .withActions(SQSActions.SendMessage)
        .withConditions(ConditionFactory.newSourceArnCondition(myTopicArn)));

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

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

Esempio di Amazon SNS

Alcuni servizi offrono condizioni aggiuntive che possono essere utilizzate nelle politiche. Amazon SNS fornisce le condizioni per consentire o negare le sottoscrizioni agli argomenti SNS in base al protocollo (ad esempio e-mail, HTTP, HTTPS Amazon SQS) e all'endpoint (ad es. indirizzo e-mail, URL, Amazon SQS ARN) della richiesta di sottoscrizione a un argomento.

```
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()));
```

Imposta il TTL JVM per le ricerche dei nomi DNS

Java Virtual Machine (JVM) memorizza nella cache le ricerche dei nomi DNS. Quando la JVM risolve un nome host in un indirizzo IP, memorizza l'indirizzo IP nella cache per un periodo di tempo specificato, noto come (TTL). time-to-live

Poiché AWS le risorse utilizzano voci di nomi DNS che cambiano occasionalmente, si consiglia di configurare la JVM con un valore TTL di 5 secondi. Questo garantisce che quando l'indirizzo IP di una risorsa cambia, l'applicazione potrà ricevere e utilizzare il nuovo indirizzo IP della risorsa richiedendo il DNS.

In alcune configurazioni Java, il TTL predefinito di JVM è impostato in modo da non aggiornare mai le voci DNS finché JVM non viene riavviato. Pertanto, se l'indirizzo IP di una AWS risorsa cambia mentre l'applicazione è ancora in esecuzione, non sarà possibile utilizzare tale risorsa finché non si riavvia manualmente la JVM e le informazioni IP memorizzate nella cache non vengono aggiornate. In questo caso, è fondamentale impostare il valore TTL della JVM in modo che aggiorni periodicamente le informazioni IP memorizzate nella cache.

Come impostare il TTL JVM

Per modificare il TTL della JVM, imposta il valore della proprietà di sicurezza networkaddress.cache.ttl, imposta la proprietà nel file per Java 8 o nel `networkaddress.cache.ttl` file per Java 11 o versioni successive. `$JAVA_HOME/jre/lib/security/java.security` `$JAVA_HOME/conf/security/java.security`

Quello che segue è un frammento di un file che mostra la cache TTL impostata su 5 secondi. `java.security`

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

Tutte le applicazioni eseguite sulla JVM rappresentata dalla variabile di `$JAVA_HOME` ambiente utilizzano questa impostazione.

Abilitazione delle metriche per AWS SDK per Java

AWS SDK per Java Possono generare metriche per la visualizzazione e il monitoraggio con Amazon CloudWatch che misurano:

- le prestazioni della tua applicazione durante l'accesso AWS

- le prestazioni del tuo JVMs quando viene utilizzato con AWS
- dettagli dell'ambiente di runtime come memoria heap, numero di thread e descrittori di file aperti

Come abilitare Java SDK Metric Generation

È necessario aggiungere la seguente dipendenza Maven per abilitare l'SDK a cui inviare le metriche. 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>
```

* Sostituisci il numero di versione con l'ultima versione dell'SDK disponibile su Maven Central.

AWS SDK per Java le metriche sono disabilitate per impostazione predefinita. Per abilitarlo per il tuo ambiente di sviluppo locale, includi una proprietà di sistema che punti al file delle credenziali AWS di sicurezza all'avvio della JVM. Ad esempio:

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

È necessario specificare il percorso del file di credenziali in modo che l'SDK possa caricare i punti dati raccolti per un'analisi successiva. CloudWatch

> **Note**
>
> Se accedi AWS da un' Amazon EC2 istanza utilizzando il servizio di metadati dell' Amazon EC2 istanza, non è necessario specificare un file di credenziali. In questo caso, devi solo specificare:
>
> ```
> -Dcom.amazonaws.sdk.enableDefaultMetrics
> ```

Tutte le metriche acquisite da si AWS SDK per Java trovano nello spazio dei nomi AWSSDK/Java e vengono caricate CloudWatch nella regione predefinita (us-east-1). Per modificare la regione, specificala utilizzando l'attributo nella proprietà di sistema. `cloudwatchRegion` Ad esempio, per impostare la CloudWatch regione su us-east-1, usa:

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

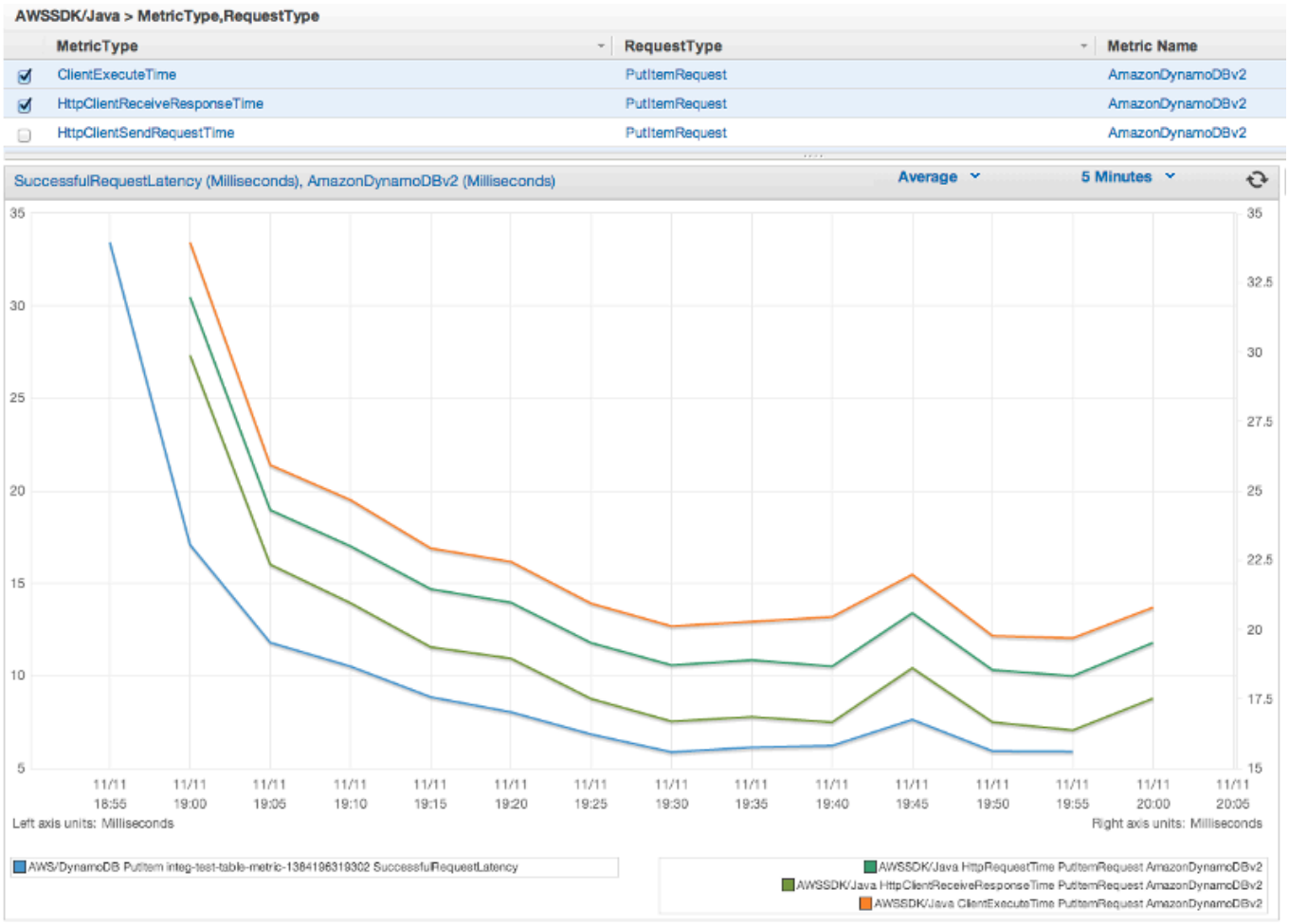
Una volta abilitata la funzionalità, ogni volta che viene inviata una richiesta di servizio, i dati metrici verranno generati AWS SDK per Java, messi in coda per il riepilogo statistico e caricati in modo asincrono circa una volta al minuto. AWS CloudWatch Una volta caricate le metriche, puoi visualizzarle utilizzando Console di gestione AWSe impostare allarmi su potenziali problemi come perdita di memoria, perdita del descrittore di file e così via.

Tipi di metriche disponibili

Il set di metriche predefinito è suddiviso in tre categorie principali:

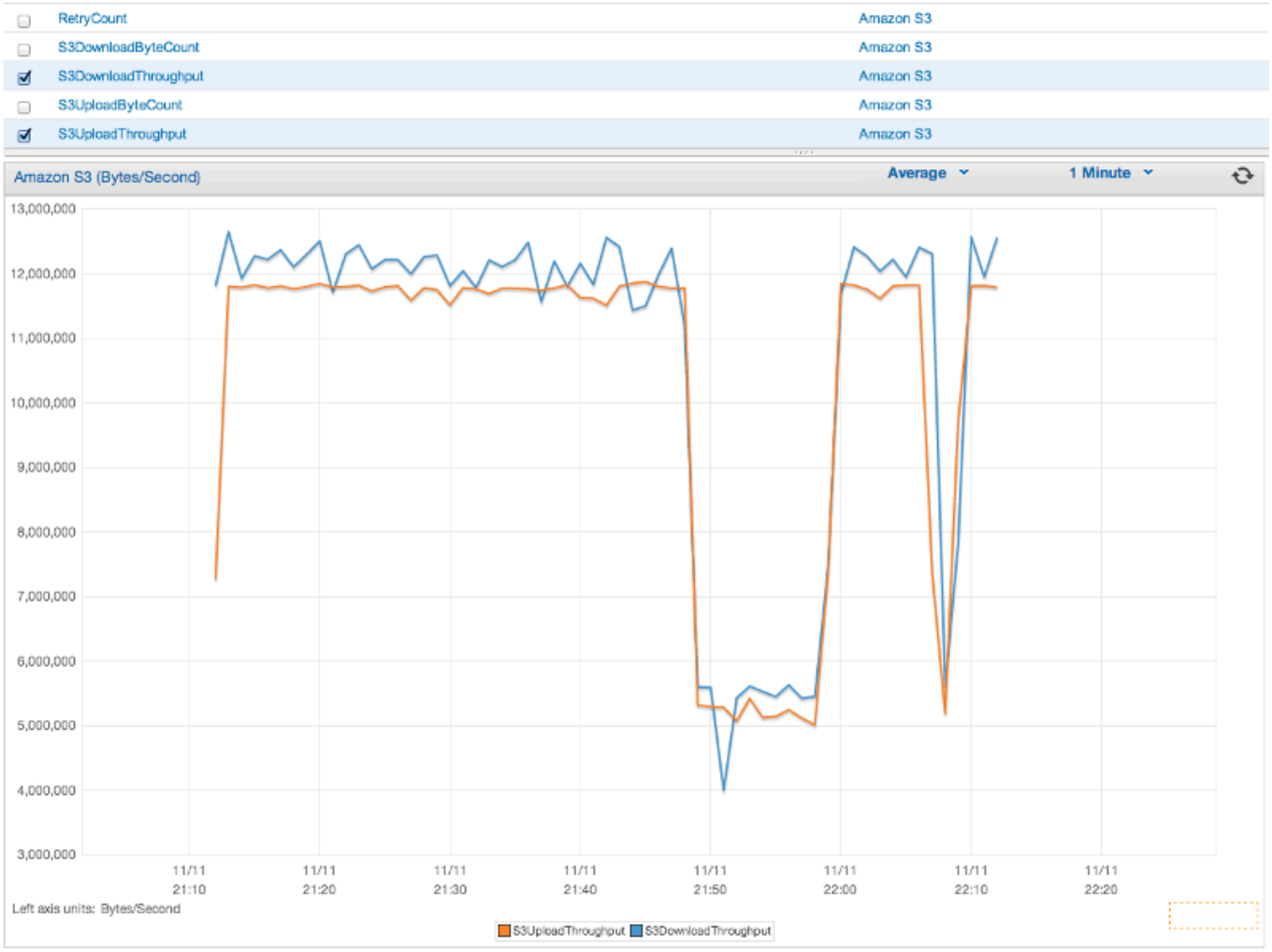
AWS Richiedi metriche

- Copre aree quali la latenza della richiesta/risposta HTTP, il numero di richieste, le eccezioni e i nuovi tentativi.



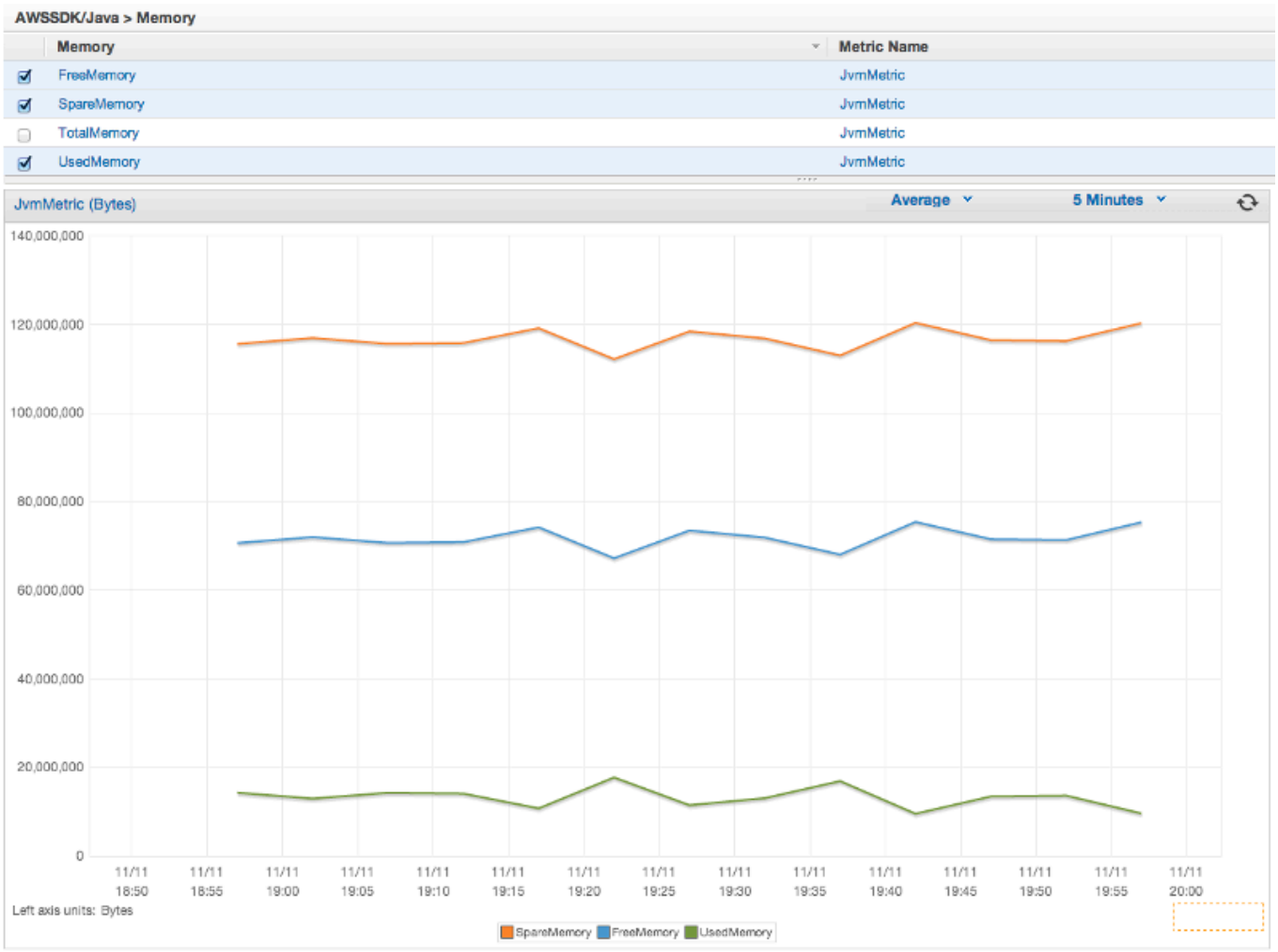
Servizio AWS Metriche

- Includi dati Servizio AWS specifici, come la velocità effettiva e il numero di byte per i caricamenti e i download di S3.



Metriche della macchina

- Copre l'ambiente di runtime, inclusi la memoria heap, il numero di thread e i descrittori di file aperti.



Se desideri escludere Machine Metrics, aggiungi `excludeMachineMetrics` alla proprietà di sistema:

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

Ulteriori informazioni

- Consulta il riepilogo del pacchetto amazonaws/metrics per un elenco completo dei tipi di metriche principali predefiniti.
- Scopri come utilizzare il file in Esempi utilizzando il CloudWatch . AWS SDK per Java CloudWatch AWS SDK per Java

- Scopri di più sull'ottimizzazione delle prestazioni nel post del blog Tuning the AWS SDK per Java to Improve Resiliency.

AWS SDK per Java Codici di esempio

Questa sezione fornisce tutorial ed esempi sull'utilizzo della AWS SDK per Java v1 per programmare i servizi. AWS

Trova il codice sorgente per questi esempi e altri nel repository degli esempi di codice di AWS documentazione su. GitHub

Per proporre un nuovo esempio di codice da far produrre al team addetto alla AWS documentazione, crea una nuova richiesta. Il team sta cercando di produrre esempi di codice che coprano scenari e casi d'uso più ampi rispetto ai semplici frammenti di codice che coprono solo le singole chiamate API. Per istruzioni, consulta le linee guida per i contributi nel repository degli esempi di codice su.. GitHub

AWS SDK per Java 2.x

Nel 2018, AWS ha rilasciato il. AWS SDK for Java 2.x Questa guida contiene istruzioni sull'utilizzo dell'SDK Java più recente insieme al codice di esempio.

> **Note**
>
> Consultate Documentazione e risorse aggiuntive per altri esempi e risorse aggiuntive disponibili per AWS SDK per Java gli sviluppatori!

CloudWatch Esempi di utilizzo di AWS SDK per Java

In questa sezione vengono forniti esempi di programmazione di CloudWatch utilizzando AWS SDK per Java.

Amazon CloudWatch monitora le tue Amazon Web Services (AWS) risorse e le applicazioni su cui esegui AWS in tempo reale. Puoi utilizzarlo CloudWatch per raccogliere e tenere traccia delle metriche, che sono variabili che puoi misurare per le tue risorse e applicazioni. CloudWatch gli allarmi inviano notifiche o apportano automaticamente modifiche alle risorse che stai monitorando in base a regole da te definite.

Per ulteriori informazioni in merito CloudWatch, consulta la Guida per l'Amazon CloudWatch utente.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ogni tecnica. Il codice di esempio completo è disponibile su GitHub. Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

Argomenti

Ottenere metriche da CloudWatch

Elencazione dei parametri

Per elencare CloudWatch le metriche, crea un metodo ListMetricsRequeste AmazonCloudWatchClient chiama `listMetrics` il. Puoi utilizzare `ListMetricsRequest` per filtrare i parametri restituiti in base a spazio dei nomi, nome parametro o dimensioni.

> **Note**
>
> Un elenco di metriche e dimensioni pubblicate dai AWS servizi è disponibile nella {https--- docs-aws-amazon-com- AmazonCloudWatch -Latest-Monitoring-CW-Support-for-AWS-html} [Amazon Metrics and Dimensions Reference] nella Guida per l'utente. CloudWatch Amazon CloudWatch

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

Le ListMetricsResultmetriche vengono restituite in un `getMetrics` chiamando il relativo metodo. I risultati possono essere paginati. Per recuperare il successivo batch di risultati, chiamate `setNextToken` l'oggetto di richiesta originale con il valore restituito dal `getNextToken` metodo dell'`ListMetricsResult`oggetto e passate l'oggetto di richiesta modificato a un'altra chiamata a. `listMetrics`

Ulteriori informazioni

- ListMetricsnell' Amazon CloudWatch API Reference.

Pubblicazione di dati dei parametri personalizzati

Alcuni AWS servizi pubblicano le proprie metriche in namespace che iniziano con "`AWS`" Puoi anche pubblicare dati metrici personalizzati utilizzando il tuo spazio dei nomi (purché non inizi con "«). `AWS`

Pubblicare dati dei parametri personalizzati

Per pubblicare i tuoi dati metrici, chiama il metodo's con a. AmazonCloudWatchClient `putMetricData` PutMetricDataRequest `PutMetricDataRequest`Devono includere lo spazio dei nomi personalizzato da utilizzare per i dati e le informazioni sul punto dati stesso in un oggetto. MetricDatum

> **Note**
>
> Non è possibile specificare uno spazio dei nomi che inizia con "». AWS I namespace che iniziano con "AWS" sono riservati all'uso da parte dei prodotti. Amazon Web Services

Importazioni

```
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;
```

Codice

```
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);
```

Ulteriori informazioni

- Utilizzo delle Amazon CloudWatch metriche nella Guida per l'utente. Amazon CloudWatch
- AWS Namespace nella Guida per l'utente. Amazon CloudWatch
- PutMetricDatanell'API Reference. Amazon CloudWatch

Lavorare con gli CloudWatch allarmi

Creazione di un allarme

Per creare un allarme basato su una CloudWatch metrica, chiama il `putMetricAlarm` metodo AmazonCloudWatchClient's PutMetricAlarmRequestcompilando le condizioni di allarme.

Importazioni

```
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;
```

Codice

```
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);
```

Elencare allarmi

Per elencare gli CloudWatch allarmi che hai creato, chiama il `describeAlarms` metodo AmazonCloudWatchClient's con un DescribeAlarmsRequestche puoi usare per impostare le opzioni relative al risultato.

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

L'elenco degli allarmi può essere ottenuto `getMetricAlarms` chiamando il comando DescribeAlarmsResultche viene restituito da. `describeAlarms`

I risultati possono essere paginati. Per recuperare il successivo batch di risultati, chiamate `setNextToken` l'oggetto di richiesta originale con il valore restituito dal `getNextToken` metodo dell'`DescribeAlarmsResult`oggetto e passate l'oggetto di richiesta modificato a un'altra chiamata a. `describeAlarms`

> **Note**
>
> Puoi anche recuperare gli allarmi per una metrica specifica utilizzando il metodo's. AmazonCloudWatchClient `describeAlarmsForMetric` L'uso è simile a `describeAlarms`.

Elimina allarmi

Per eliminare gli CloudWatch allarmi, chiama il `deleteAlarms` metodo AmazonCloudWatchClient's con uno DeleteAlarmsRequesto più nomi di allarmi che desideri eliminare.

Importazioni

```
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;
```

Codice

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

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

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

Ulteriori informazioni

- Creazione di Amazon CloudWatch allarmi nella Guida per l'utente Amazon CloudWatch
- PutMetricAlarmnell' Amazon CloudWatch API Reference
- DescribeAlarmsnell' Amazon CloudWatch API Reference
- DeleteAlarmsnell' Amazon CloudWatch API Reference

Utilizzo delle azioni di allarme in CloudWatch

Utilizzando le azioni di CloudWatch allarme, è possibile creare allarmi che eseguono azioni come l'arresto automatico, la chiusura, il riavvio o il ripristino delle istanze. Amazon EC2

> **Note**
>
> Le azioni di allarme possono essere aggiunte a un allarme utilizzando il metodo's durante la PutMetricAlarmRequestcreazione di un allarme. `setAlarmActions`

Attivare le operazioni di allarme

Per abilitare le azioni di allarme per un CloudWatch allarme, chiama AmazonCloudWatchClient th's `enableAlarmActions` EnableAlarmActionsRequestinserendo uno o più nomi di allarmi di cui desideri abilitare le azioni.

Importazioni

```
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;
```

Codice

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

EnableAlarmActionsRequest request = new EnableAlarmActionsRequest()
```

```
    .withAlarmNames(alarm);

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

Disattivare le operazioni di allarme

Per disabilitare le azioni di CloudWatch allarme di un allarme, AmazonCloudWatchClient chiama gli avvisi `disableAlarmActions` con DisableAlarmActionsRequestuno o più nomi di allarmi di cui desideri disabilitare le azioni.

Importazioni

```
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;
```

Codice

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

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

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

Ulteriori informazioni

- Crea allarmi per interrompere, terminare, riavviare o ripristinare un'istanza nella Guida per l'utente Amazon CloudWatch
- PutMetricAlarmnel riferimento all'API Amazon CloudWatch
- EnableAlarmActionsnell' Amazon CloudWatch API Reference
- DisableAlarmActionsnell' Amazon CloudWatch API Reference

Invio di eventi a CloudWatch

CloudWatch Events fornisce un flusso quasi in tempo reale di eventi di sistema che descrivono le modifiche AWS delle risorse a Amazon EC2 istanze, Lambda funzioni, Kinesis flussi, Amazon ECS

attività, macchine a Step Functions stati, Amazon SNS argomenti, Amazon SQS code o destinazioni integrate. Puoi abbinare gli eventi e instradarli verso una o più funzioni o stream target utilizzando regole semplici.

Aggiunta di eventi

Per aggiungere CloudWatch eventi personalizzati, chiamate il `putEvents` metodo AmazonCloudWatchEventsClient's con un PutEventsRequestoggetto che contiene uno o più PutEventsRequestEntryoggetti che forniscono dettagli su ogni evento. Puoi specificare diversi parametri per la voce, ad esempio l'origine e il tipo di evento, le risorse associate all'evento e così via.

> **Note**
>
> Puoi specificare un massimo di 10 eventi per chiamata a `putEvents`.

Importazioni

```
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;
```

Codice

```
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);
```

Aggiunta di regole

Per creare o aggiornare una regola, chiamate il `putRule` metodo AmazonCloudWatchEventsClient the's PutRuleRequestcon a con il nome della regola e parametri facoltativi come il modello di evento, il IAM ruolo da associare alla regola e un'espressione di pianificazione che descriva la frequenza di esecuzione della regola.

Importazioni

```
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;
```

Codice

```
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);
```

Aggiunta di target

I target sono le risorse che vengono invocate quando una regola viene attivata. Gli obiettivi di esempio includono Amazon EC2 istanze, Lambda funzioni, Kinesis flussi, Amazon ECS attività, macchine a Step Functions stati e destinazioni integrate.

Per aggiungere un obiettivo a una regola, chiama il `putTargets` metodo AmazonCloudWatchEventsClient's PutTargetsRequestcontenente la regola da aggiornare e un elenco di obiettivi da aggiungere alla regola.

Importazioni

```
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;
```

Codice

```
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);
```

Ulteriori informazioni

- Aggiungere eventi PutEvents nella Guida per l' Amazon CloudWatch Events utente
- Pianifica le espressioni per le regole nella Guida per l' Amazon CloudWatch Events utente
- Tipi di CloudWatch eventi per gli eventi indicati nella Guida per l' Amazon CloudWatch Events utente
- Eventi e modelli di eventi nella Guida Amazon CloudWatch Events per l'utente
- PutEventsnell' Amazon CloudWatch Events API Reference
- PutTargetsnell' Amazon CloudWatch Events API Reference
- PutRulenell' Amazon CloudWatch Events API Reference

DynamoDB Esempi di utilizzo di AWS SDK per Java

In questa sezione vengono forniti esempi di programmazione di DynamoDB utilizzando AWS SDK per Java.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ogni tecnica. Il codice di esempio completo è disponibile su GitHub. Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

Argomenti

Usa AWS endpoint basati su account

DynamoDB AWS offre endpoint basati su account che possono migliorare le prestazioni utilizzando l'ID dell'account per semplificare AWS il routing delle richieste.

Per sfruttare questa funzionalità, è necessario utilizzare la versione 1.12.771 o successiva della versione 1 di. AWS SDK per JavaPuoi trovare l'ultima versione dell'SDK elencata nell'archivio centrale di Maven. Dopo che una versione supportata di SDK è attiva, utilizza automaticamente i nuovi endpoint.

Se desideri disattivare il routing basato sull'account, hai quattro opzioni:

- Configurare un client di servizio DynamoDB con `AccountIdEndpointMode` l'impostazione su. `DISABLED`
- Imposta una variabile di ambiente.
- Imposta una proprietà del sistema JVM.
- Aggiorna l'impostazione del file di AWS configurazione condiviso.

Il seguente frammento è un esempio di come disabilitare il routing basato su account configurando un client di servizio DynamoDB:

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

```
AWSCredentialsProvider credentialsProvider = new
 EnvironmentVariableCredentialsProvider();

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

La AWS SDKs and Tools Reference Guide fornisce ulteriori informazioni sulle ultime tre opzioni di configurazione.

Lavorare con le tabelle in DynamoDB

Le tabelle sono i contenitori per tutti gli elementi di un DynamoDB database. Prima di poter aggiungere o rimuovere dati da DynamoDB, è necessario creare una tabella.

Per ogni tabella, devi definire:

- Un nome di tabella univoco per l'account e la regione.
- Una chiave primaria per la quale ogni valore deve essere univoco; due item nella tabella non possono avere lo stesso valore della chiave primaria.

  La chiave primaria può essere semplice, costituita da una singola chiave di partizione (HASH), o composita, costituita da una chiave di partizione e una di ordinamento (RANGE).

  A ogni valore chiave è associato un tipo di dati, enumerato dalla classe. ScalarAttributeType Il valore della chiave può essere binario (B), numerico (N) o una stringa (S). Per ulteriori informazioni, consulta Regole di denominazione e tipi di dati nella Guida per gli sviluppatori. Amazon DynamoDB
- Valori di throughput assegnati che definiscono il numero di unità di capacità di lettura/scrittura riservate per la tabella.

  > **Note**
  >
  > Amazon DynamoDB i prezzi si basano sui valori di throughput assegnati che imposti sulle tabelle, quindi prenota solo la capacità che ritieni necessaria per la tabella.

Il throughput assegnato per una tabella può essere modificato in qualsiasi momento, in modo da poter regolare la capacità in caso di variazioni delle esigenze.

Creazione di una tabella

Usa il `createTable` metodo del DynamoDB client per creare una nuova DynamoDB tabella. È necessario costruire gli attributi della tabella e uno schema della tabella, entrambi utilizzati per identificare la chiave primaria della tabella. Inoltre, occorre fornire i valori del throughput assegnato iniziali e un nome della tabella. Definisci solo gli attributi chiave della tabella durante la creazione della DynamoDB tabella.

> **Note**
>
> Se esiste già una tabella con il nome scelto, ne AmazonServiceExceptionviene generata una.

Importazioni

```
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;
```

Creazione di una tabella con una chiave primaria semplice

Questo codice consente di creare una tabella con una chiave primaria semplice ("Name").

Codice

```
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);
}
```

Vedi l'esempio completo su. GitHub

Creazione di una tabella con una chiave primaria composita

Aggiungi un altro AttributeDefinitione KeySchemaElementa CreateTableRequest.

Codice

```
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);
```

Guarda l'esempio completo su GitHub.

Elencare tabelle

È possibile elencare le tabelle in una particolare regione chiamando il `listTables` metodo del DynamoDB client.

> **Note**
>
> Se la tabella denominata non esiste per il tuo account e la tua regione, ResourceNotFoundExceptionviene generata a.

Importazioni

```
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;
```

Codice

```
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;
```

```
        }
```

Per impostazione predefinita, vengono restituite fino a 100 tabelle per chiamata: da utilizzare `getLastEvaluatedTableName` sull'ListTablesResultoggetto restituito per ottenere l'ultima tabella valutata. Puoi utilizzare questo valore per avviare la visualizzazione dell'elenco dopo l'ultimo valore restituito dalla visualizzazione dell'elenco precedente.

Vedi l'esempio completo su. GitHub

Descrizione di (recupero delle informazioni su) una tabella

Chiama il `describeTable` metodo del DynamoDB client.

> **Note**
>
> Se la tabella denominata non esiste per il tuo account e la tua regione, ResourceNotFoundExceptionviene generata a.

Importazioni

```
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;
```

Codice

```
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);
}
```

Vedi l'esempio completo su. GitHub

Modifica (aggiornamento) di una tabella

Puoi modificare i valori di throughput assegnati alla tabella in qualsiasi momento chiamando il metodo del DynamoDBupdateTableclient.

> **Note**
>
> Se la tabella denominata non esiste per il tuo account e la tua regione, ResourceNotFoundExceptionviene generata a.

Importazioni

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

Codice

```
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);
}
```

Vedi l'esempio completo su. GitHub

Eliminazione di una tabella

Chiama il `deleteTable` metodo del DynamoDB client e passagli il nome della tabella.

> **Note**
>
> Se la tabella denominata non esiste per il tuo account e la tua regione, ResourceNotFoundExceptionviene generata a.

Importazioni

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

Codice

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

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

Vedi l'esempio completo su. GitHub

Ulteriori informazioni

- Linee guida per l'utilizzo delle tabelle nella Guida per Amazon DynamoDB gli sviluppatori
- Utilizzo delle tabelle DynamoDB nella Guida per gli Amazon DynamoDB sviluppatori

Utilizzo degli elementi in DynamoDB

In DynamoDB, un elemento è una raccolta di attributi, ognuno dei quali ha un nome e un valore. Un valore attributo può essere un tipo scalare, set o documento. Per ulteriori informazioni, consulta Regole di denominazione e tipi di dati nella Guida per gli Amazon DynamoDB sviluppatori.

Recuperare (ottenere) un item da una tabella

Chiama il `getItem` metodo del AmazonDynamo DB e passagli un GetItemRequestoggetto con il nome della tabella e il valore della chiave primaria dell'elemento che desideri. Restituisce un GetItemResultoggetto.

È possibile utilizzare il `getItem()` metodo dell'`GetItemResult`oggetto restituito per recuperare una mappa di coppie chiave (String AttributeValue) e valore () associate all'elemento.

Importazioni

```
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;
```

Codice

```
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);
```

Vedi l'esempio completo su. GitHub

Aggiunta di un nuovo item a una tabella

Creare una mappa di coppie chiave-valore che rappresentino gli attributi della voce. Queste devono includere valori per i campi chiave primaria della tabella. Se l'elemento identificato dalla chiave primaria esiste già, i relativi campi vengono aggiornati dalla richiesta.

> **Note**
>
> Se la tabella denominata non esiste per il tuo account e la tua regione, ResourceNotFoundExceptionviene generata a.

Importazioni

```
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;
```

Codice

```
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);
```

Vedi l'esempio completo su. GitHub

Aggiornamento di un item esistente in una tabella

È possibile aggiornare un attributo per un elemento già esistente in una tabella utilizzando il `updateItem` metodo del AmazonDynamo DB, fornendo un nome di tabella, un valore della chiave primaria e una mappa di campi da aggiornare.

> **Note**
>
> Se la tabella denominata non esiste per il tuo account e la tua regione, o se l'elemento identificato dalla chiave primaria che hai passato non esiste, ResourceNotFoundExceptionviene generato a.

Importazioni

```
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;
```

Codice

```
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);
```

Vedi l'esempio completo su. GitHub

Usa la classe Dynamo DBMapper

AWS SDK per JavaFornisce una DBMapper classe Dynamo, che consente di mappare le classi lato client alle tabelle. Amazon DynamoDB Per utilizzare la DBMapper classe Dynamo, definite la relazione tra gli elementi di una DynamoDB tabella e le istanze di oggetto corrispondenti nel codice utilizzando le annotazioni (come mostrato nel seguente esempio di codice). La DBMapper classe Dynamo consente di accedere alle tabelle, eseguire varie operazioni di creazione, lettura, aggiornamento ed eliminazione (CRUD) ed eseguire query.

> **Note**
>
> La DBMapper classe Dynamo non consente di creare, aggiornare o eliminare tabelle.

Importazioni

```
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;
```

Codice

Il seguente esempio di codice Java mostra come aggiungere contenuti alla tabella Music utilizzando la classe DBMapperDynamo. Dopo aver aggiunto il contenuto alla tabella, notate che un elemento

viene caricato utilizzando i tasti Partition e Sort. Quindi la voce Awards viene aggiornata. Per informazioni sulla creazione della tabella Music, consulta Create a Table nella Amazon DynamoDB Developer Guide.

```
        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;
    }
}
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Linee guida per l'utilizzo degli elementi nella Guida per gli Amazon DynamoDB sviluppatori
- Utilizzo degli elementi contenuti DynamoDB nella Guida per gli Amazon DynamoDB sviluppatori

Amazon EC2 Esempi di utilizzo di AWS SDK per Java

Questa sezione fornisce esempi di programmazione Amazon EC2con AWS SDK per Java.

Argomenti

- Tutorial: avvio di un' EC2 istanza
- Utilizzo dei ruoli IAM per concedere l'accesso alle AWS risorse su Amazon EC2
- Tutorial: istanze Amazon EC2 Spot
- Tutorial: Gestione avanzata delle richieste Amazon EC2 Spot
- Gestione delle Amazon EC2 istanze
- Utilizzo di indirizzi IP elastici in Amazon EC2
- Usa aree e zone di disponibilità
- Lavorare con coppie di Amazon EC2 chiavi
- Lavorare con i gruppi di sicurezza in Amazon EC2

Tutorial: avvio di un' EC2 istanza

Questo tutorial mostra come utilizzare per AWS SDK per Java avviare un' EC2 istanza.

Argomenti

- Prerequisiti
- Creare un gruppo Amazon EC2 di sicurezza
- Crea una coppia di chiavi
- Esegui un' Amazon EC2 istanza

Prerequisiti

Prima di iniziare, assicurati di aver creato un file Account AWS e di aver impostato AWS le tue credenziali. Per ulteriori informazioni, consulta Nozioni di base su .

Creare un gruppo Amazon EC2 di sicurezza

EC2-Classic sta per andare in pensione

> **Warning**
>
> Ritireremo EC2 -Classic il 15 agosto 2022. Ti consigliamo di migrare da EC2 -Classic a un VPC. Per ulteriori informazioni, consulta il post del blog EC2-Classic-Classic Networking is Retiring — Ecco come prepararsi.

Crea un gruppo di sicurezza, che funge da firewall virtuale che controlla il traffico di rete per una o più istanze. EC2 Per impostazione predefinita, Amazon EC2 associa le istanze a un gruppo di sicurezza che non consente il traffico in entrata. Puoi creare un gruppo di sicurezza che consenta alle EC2 istanze di accettare un determinato traffico. Ad esempio, se è necessario connettersi a un'istanza Linux, è necessario configurare il gruppo di sicurezza per consentire il traffico SSH. È possibile creare un gruppo di sicurezza utilizzando la Amazon EC2 console o il AWS SDK per Java.

È possibile creare un gruppo di sicurezza da utilizzare in EC2 -Classic o EC2 -VPC. Per ulteriori informazioni su EC2 -Classic e EC2 -VPC, consulta Supported Platforms nella Amazon EC2 User Guide for Linux Instances.

Per ulteriori informazioni sulla creazione di un gruppo di sicurezza utilizzando la Amazon EC2 console, consulta Amazon EC2 Security Groups nella Guida Amazon EC2 utente per le istanze Linux.

1. Crea e inizializza un'CreateSecurityGroupRequestistanza. Utilizzare il withGroupNamemetodo per impostare il nome del gruppo di sicurezza e il metodo WithDescription per impostare la descrizione del gruppo di sicurezza, come segue:

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

Il nome del gruppo di sicurezza deve essere univoco all'interno della AWS regione in cui si inizializza il Amazon EC2 client. È necessario utilizzare caratteri US-ASCII per il nome e la descrizione del gruppo di sicurezza.

2. Passate l'oggetto della richiesta come parametro al createSecurityGroupmetodo. Il metodo restituisce un CreateSecurityGroupResultoggetto, come segue:

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

Se si tenta di creare un gruppo di sicurezza con lo stesso nome di un gruppo di sicurezza esistente, `createSecurityGroup` genera un'eccezione.

Per impostazione predefinita, un nuovo gruppo di sicurezza non consente alcun traffico in entrata verso l' Amazon EC2 istanza. Per consentire il traffico in entrata, devi autorizzare esplicitamente l'ingresso del gruppo di sicurezza. È possibile autorizzare l'ingresso per singoli indirizzi IP, per un intervallo di indirizzi IP, per un protocollo specifico e per le porte TCP/UDP.

1. Crea e inizializza un'istanza. IpPermission Utilizzate il metodo WithIPv4Ranges per impostare l'intervallo di indirizzi IP per cui autorizzare l'ingresso e utilizzate il metodo per impostare il withIpProtocolprotocollo IP. Utilizzate i withToPortmetodi withFromPortand per specificare l'intervallo di porte per cui autorizzare l'ingresso, come segue:

```
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);
```

Tutte le condizioni specificate nell'`IpPermission`oggetto devono essere soddisfatte per consentire l'ingresso.

Specificare l'indirizzo IP utilizzando la notazione CIDR. Se si specifica il protocollo come TCP/UDP, è necessario fornire una porta di origine e una porta di destinazione. È possibile autorizzare le porte solo se si specifica TCP o UDP.

2. Crea e inizializza un'istanza. AuthorizeSecurityGroupIngressRequest Utilizzate il `withGroupName` metodo per specificare il nome del gruppo di sicurezza e passate al withIpPermissionsmetodo l'`IpPermission`oggetto inizializzato in precedenza, come segue:

```
AuthorizeSecurityGroupIngressRequest authorizeSecurityGroupIngressRequest =
    new AuthorizeSecurityGroupIngressRequest();

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

3. Passate l'oggetto della richiesta al metodo authorizeSecurityGroupIngress, come segue:

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

Se chiamate `authorizeSecurityGroupIngress` con indirizzi IP per i quali l'ingresso è già autorizzato, il metodo genera un'eccezione. Crea e inizializza un nuovo `IpPermission` oggetto per autorizzare l'ingresso per diverse porte e protocolli IPs prima della chiamata. `AuthorizeSecurityGroupIngress`

Ogni volta che chiamate i metodi authorizeSecurityGroupIngress o authorizeSecurityGroupEgress, viene aggiunta una regola al gruppo di sicurezza.

Crea una coppia di chiavi

È necessario specificare una coppia di chiavi all'avvio di un' EC2 istanza e quindi specificare la chiave privata della coppia di chiavi quando ci si connette all'istanza. Puoi creare una coppia di chiavi o utilizzare una coppia di chiavi esistente che hai usato per avviare altre istanze. Per ulteriori informazioni, consulta Amazon EC2 Key Pairs nella Guida per l' Amazon EC2 utente delle istanze Linux.

1. Crea e inizializza un'CreateKeyPairRequestistanza. Utilizzate il withKeyNamemetodo per impostare il nome della coppia di key pair, come segue:

```
CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();
```

```
createKeyPairRequest.withKeyName(keyName);
```

> **Important**
>
> I nomi delle coppie di chiavi devono essere univoci. Se tenti di creare una coppia di chiavi con lo stesso nome di una coppia di chiavi esistente, otterrai un'eccezione.

2. Passa l'oggetto della richiesta al createKeyPairmetodo. Il metodo restituisce un'CreateKeyPairResultistanza, come segue:

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

3. Chiamate il getKeyPairmetodo dell'oggetto risultato per ottenere un KeyPairoggetto. Chiamate il getKeyMaterialmetodo dell'`KeyPair`oggetto per ottenere la chiave privata non crittografata con codifica PEM, come segue:

```
KeyPair keyPair = new KeyPair();

keyPair = createKeyPairResult.getKeyPair();

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

Esegui un' Amazon EC2 istanza

Utilizza la seguente procedura per avviare una o più EC2 istanze configurate in modo identico dalla stessa Amazon Machine Image (AMI). Dopo aver creato le EC2 istanze, puoi verificarne lo stato. Dopo l'esecuzione EC2 delle istanze, puoi connetterti ad esse.

1. Crea e inizializza un'RunInstancesRequestistanza. Assicurati che l'AMI, la key pair e il gruppo di sicurezza che specifichi esistano nella regione specificata quando hai creato l'oggetto client.

```
RunInstancesRequest runInstancesRequest =
   new RunInstancesRequest();

runInstancesRequest.withImageId("ami-a9d09ed1")
                   .withInstanceType(InstanceType.T1Micro)
                   .withMinCount(1)
                   .withMaxCount(1)
```

```
                    .withKeyName("my-key-pair")
                    .withSecurityGroups("my-security-group");
```

withImageId

- L'ID dell'AMI. Per scoprire come trovare un AMIs servizio pubblico fornito da Amazon o crearne uno personalizzato, consulta Amazon Machine Image (AMI).

withInstanceType

- Un tipo di istanza compatibile con l'AMI specificata. Per ulteriori informazioni, consulta Tipi di istanze nella Guida Amazon EC2 utente per le istanze Linux.

withMinCount

- Il numero minimo di EC2 istanze da avviare. Se si tratta di un numero di istanze superiore a quello che è Amazon EC2 possibile avviare nella zona di disponibilità di destinazione, non Amazon EC2 viene avviata alcuna istanza.

withMaxCount

- Il numero massimo di EC2 istanze da avviare. Se si tratta di un numero di istanze superiore a quello che è Amazon EC2 possibile avviare nella zona di disponibilità di destinazione, Amazon EC2 avvia il maggior numero possibile di istanze sopra elencate. `MinCount` È possibile avviare tra 1 e il numero massimo di istanze consentite per il tipo di istanza. Per ulteriori informazioni, consulta Quante istanze posso eseguire Amazon EC2 nelle Domande frequenti generali. Amazon EC2

withKeyName

- Il nome della EC2 key pair. Se l'istanza viene avviata senza specificare una coppia di chiavi, non potrai connetterti a essa. Per ulteriori informazioni, consultare Creare una coppia di chiavi.

withSecurityGroups

- Uno o più gruppi di sicurezza. Per ulteriori informazioni, consulta Creare un gruppo Amazon EC2 di sicurezza.

2. Avvia le istanze passando l'oggetto di richiesta al metodo RunInstances. Il metodo restituisce un RunInstancesResultoggetto, come segue:

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

Una volta che l'istanza è in esecuzione, puoi connetterti ad essa utilizzando la tua key pair. Per ulteriori informazioni, consulta Connect to Your Linux Instances. nella Guida per l' Amazon EC2 utente delle istanze Linux.

Utilizzo dei ruoli IAM per concedere l'accesso alle AWS risorse su Amazon EC2

Tutte le richieste a Amazon Web Services (AWS) devono essere firmate crittograficamente utilizzando credenziali emesse da. AWS Puoi utilizzare i ruoli IAM per garantire comodamente un accesso sicuro alle AWS risorse delle tue istanze. Amazon EC2

Questo argomento fornisce informazioni su come utilizzare i ruoli IAM con le applicazioni Java SDK in esecuzione. Amazon EC2 Per ulteriori informazioni sulle istanze IAM, consulta IAM Roles for Amazon EC2 nella Guida Amazon EC2 utente per le istanze Linux.

La catena di provider e EC2 i profili di istanza predefiniti

Se l'applicazione crea un AWS client utilizzando il costruttore predefinito, il client cercherà le credenziali utilizzando la catena di provider di credenziali di default, nell'ordine seguente:

1. Nelle proprietà del sistema Java: `aws.accessKeyId` e `aws.secretKey`.
2. Nelle variabili di ambiente del sistema: `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY`.
3. Nel file delle credenziali predefinito (il percorso di questo file varia in base alla piattaforma).
4. Credenziali fornite tramite il servizio Amazon EC2 contenitore se la variabile di `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` ambiente è impostata e il responsabile della sicurezza è autorizzato ad accedere alla variabile.
5. Nel profilo di istanza, le credenziali presenti nei metadati dell'istanza associati al ruolo IAM per l'istanza. EC2
6. Credenziali Web Identity Token dall'ambiente o dal contenitore.

La fase relativa alle credenziali del profilo di istanza nella catena di provider predefinita è disponibile solo quando si esegue l'applicazione su un' Amazon EC2 istanza, ma offre la massima facilità d'uso e la massima sicurezza quando si lavora con Amazon EC2 le istanze. Puoi anche passare un'InstanceProfileCredentialsProvideristanza direttamente al costruttore del client per ottenere le credenziali del profilo dell'istanza senza procedere attraverso l'intera catena di provider predefinita.

Per esempio:

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

Quando si utilizza questo approccio, l'SDK recupera AWS credenziali temporanee con le stesse autorizzazioni di quelle associate al ruolo IAM associato all'istanza nel relativo profilo di istanza. Amazon EC2 Sebbene queste credenziali siano temporanee e alla fine scadano, le aggiorna `InstanceProfileCredentialsProvider` periodicamente in modo che le credenziali ottenute continuino a consentire l'accesso a. AWS

> **Important**
>
> L'aggiornamento automatico delle credenziali avviene solo quando si utilizza il costruttore client predefinito, che ne crea uno proprio `InstanceProfileCredentialsProvider` come parte della catena di provider predefinita, o quando si passa un'`InstanceProfileCredentialsProvider`istanza direttamente al costruttore del client. Se utilizzi un altro metodo per ottenere o passare le credenziali del profilo dell'istanza, sei responsabile del controllo e dell'aggiornamento delle credenziali scadute.

Se il costruttore del client non riesce a trovare le credenziali utilizzando la catena di fornitori di credenziali, genererà un. AmazonClientException

Procedura dettagliata: utilizzo dei ruoli IAM per le istanze EC2

La seguente procedura dettagliata mostra come recuperare un oggetto Amazon S3 utilizzando un ruolo IAM per gestire l'accesso.

Creazione di un ruolo IAM

Crea un ruolo IAM che garantisca l'accesso in sola lettura a. Amazon S3

1. Apri la console IAM.
2. Nel riquadro di navigazione, seleziona Ruoli, quindi Crea nuovo ruolo.
3. Inserisci un nome per il ruolo, quindi seleziona Next Step (Fase successiva). Ricorda questo nome, poiché ti servirà all'avvio dell' Amazon EC2 istanza.
4. Nella pagina Seleziona il tipo di ruolo, in Servizio AWS Ruoli, seleziona Amazon EC2 .

5. Nella pagina Imposta autorizzazioni, in Seleziona modello di policy, seleziona Accesso in sola Amazon S3 lettura, quindi Passaggio successivo.
6. Nella pagina di revisione, seleziona Crea ruolo.

Avvia un' EC2 istanza e specifica il tuo ruolo IAM

Puoi avviare un' Amazon EC2 istanza con un ruolo IAM utilizzando la Amazon EC2 console o il AWS SDK per Java.

- Per avviare un' Amazon EC2 istanza utilizzando la console, segui le istruzioni riportate in Getting Started with Amazon EC2 Linux Instances nella Guida per l' Amazon EC2 utente delle istanze Linux.

  Quando raggiungi la pagina Review Instance Launch (Verifica del lancio dell'istanza), seleziona Edit instance details (Modifica dettagli istanza). Nel ruolo IAM, scegli il ruolo IAM che hai creato in precedenza. Completa la procedura come descritto.

  > **Note**
  >
  > Dovrai creare o utilizzare un gruppo di sicurezza e una coppia di chiavi esistenti per connetterti all'istanza.

- Per avviare un' Amazon EC2 istanza con un ruolo IAM utilizzando il AWS SDK per Java, consulta Run an Amazon EC2 Instance.

Crea la tua applicazione

Creiamo l'applicazione di esempio da eseguire sull' EC2 istanza. Per prima cosa, crea una directory che puoi usare per contenere i file del tutorial (ad esempio,`GetS3ObjectApp`).

Quindi, copiate le AWS SDK per Java librerie nella cartella appena creata. Se le hai AWS SDK per Java scaricate nella tua `~/Downloads` directory, puoi copiarle usando i seguenti comandi:

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

Aprite un nuovo file, richiamatelo `GetS3Object.java` e aggiungete il seguente codice:

```
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();
  }
}
```

Apri un nuovo file, chiamalo `build.xml` e aggiungi le seguenti righe:

```
<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>
```

Compila ed esegui il programma modificato. Nota che non ci sono credenziali memorizzate nel programma. Pertanto, a meno che non siano già state specificate AWS le credenziali, il codice verrà generato. `AmazonServiceException` Per esempio:

```
$ 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
```

Trasferisci il programma compilato sulla tua istanza EC2

Trasferisci il programma sulla tua Amazon EC2 istanza utilizzando secure copy (), insieme alle AWS SDK per Java librerie. La sequenza di comandi è simile alla seguente.

```
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**
>
> A seconda della distribuzione Linux utilizzata, il nome utente potrebbe essere «ec2-user», «root» o «ubuntu». Per ottenere il nome DNS pubblico dell'istanza, apri la EC2 console e cerca il valore Public DNS nella scheda Descrizione (ad esempio,). `ec2-198-51-100-1.compute-1.amazonaws.com`

Nei comandi precedenti:

- `GetS3Object.class`è il tuo programma compilato
- `build.xml`è il file ant utilizzato per creare ed eseguire il programma
- le `third-party` directory `lib` e sono le cartelle della libreria corrispondenti da. AWS SDK per Java
- L'`-r`interruttore indica che `scp` dovrebbe fare una copia ricorsiva di tutti i contenuti delle `third-party` directory `library` e della distribuzione. AWS SDK per Java
- L'`-p`interruttore indica che `scp` deve conservare le autorizzazioni dei file di origine quando li copia nella destinazione.

> **Note**
>
> Lo `-p` switch funziona solo su Linux, macOS o Unix. Se stai copiando file da Windows, potresti dover correggere le autorizzazioni relative ai file sull'istanza utilizzando il seguente comando:

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

Esegui il programma di esempio sull'istanza EC2

Per eseguire il programma, connettiti alla tua Amazon EC2 istanza. Per ulteriori informazioni, consulta Connect to Your Linux Instances nella Amazon EC2 User Guide for Linux Instances.

Se non **`ant`** è disponibile sulla tua istanza, installala utilizzando il seguente comando:

```
sudo yum install ant
```

Quindi, esegui il programma `ant` nel modo seguente:

```
ant run
```

Il programma scriverà il contenuto dell' Amazon S3 oggetto nella finestra di comando.

Tutorial: istanze Amazon EC2 Spot

Panoramica

Le istanze Spot consentono di fare offerte sulla capacità inutilizzata Amazon Elastic Compute Cloud (Amazon EC2) fino al 90% rispetto al prezzo delle istanze on demand e di eseguire le istanze acquisite finché l'offerta supera il prezzo Spot corrente. Amazon EC2 modifica periodicamente il prezzo Spot in base alla domanda e all'offerta e i clienti le cui offerte lo soddisfano o superano ottengono l'accesso alle istanze Spot disponibili. Come le istanze on demand e le istanze riservate, le istanze Spot offrono un'altra opzione per ottenere una maggiore capacità di elaborazione.

Le istanze Spot possono ridurre in modo significativo i Amazon EC2 costi per l'elaborazione in batch, la ricerca scientifica, l'elaborazione delle immagini, la codifica video, la scansione di dati e web, l'analisi finanziaria e i test. Inoltre, le istanze Spot consentono di accedere a grandi quantità di capacità aggiuntiva in situazioni in cui la necessità di tale capacità non è urgente.

Per utilizzare le istanze Spot, farne richiesta specificando il prezzo massimo che si desidera pagare per ora di istanza; questa sarà la tua offerta. Se la tua offerta supera il prezzo Spot corrente, la tua richiesta è soddisfatta e le tue istanze saranno eseguite finché non sceglierai di terminarle o finché il prezzo Spot non supererà nuovamente la tua offerta (a seconda di quale dei due si verifica prima).

È importante notare che:

- Spesso pagherai meno all'ora rispetto alla tua offerta. Amazon EC2 aggiusta periodicamente il prezzo spot in base all'arrivo delle richieste e alle variazioni dell'offerta disponibile. Tutti devono

pagare lo stesso prezzo Spot per quel periodo, indipendentemente dal fatto che la loro offerta fosse più alta. Pertanto, potresti pagare meno, ma non pagherai mai di più rispetto alla tua offerta.

- Se utilizzi istanze Spot e la tua offerta non soddisfa o supera più il prezzo Spot corrente, le istanze verranno chiuse. Ciò significa che dovrai assicurarti che i tuoi carichi di lavoro e le tue applicazioni siano sufficientemente flessibili da sfruttare questa capacità opportunistica.

Le istanze Spot si comportano esattamente come le altre Amazon EC2 istanze durante l'esecuzione e, come altre Amazon EC2 istanze, le istanze Spot possono essere terminate quando non sono più necessarie. Se termini la tua istanza, paghi per la frazione di ora utilizzata (come faresti per le istanze On demand o Riservate). Tuttavia, se il prezzo Spot supera l'offerta e l'istanza viene chiusa entro il termine Amazon EC2, non ti verrà addebitata alcuna ora parziale di utilizzo.

Questo tutorial mostra come utilizzare per AWS SDK per Java eseguire le seguenti operazioni.

- Invia una richiesta Spot
- Stabilisci quando la richiesta Spot viene soddisfatta
- Annulla la richiesta Spot
- Terminare le istanze associate

Prerequisiti

Per utilizzare questo tutorial è necessario averlo AWS SDK per Java installato, oltre a soddisfare i prerequisiti di installazione di base. Per ulteriori informazioni, consulta Configurare il AWS SDK per Java.

Fase 1: Configurazione delle credenziali

Per iniziare a utilizzare questo esempio di codice, è necessario impostare AWS le credenziali. Per istruzioni su come eseguire questa operazione, consulta Configurare le AWS credenziali e la regione per lo sviluppo.

> **Note**
>
> Ti consigliamo di utilizzare le credenziali di un utente IAM per fornire questi valori. Per ulteriori informazioni, consulta Registrazione AWS e creazione di un utente IAM.

Ora che hai configurato le impostazioni, puoi iniziare a utilizzare il codice riportato nell'esempio.

Fase 2: Configurazione di un gruppo di sicurezza

Un gruppo di sicurezza funge da firewall che controlla il traffico consentito in entrata e in uscita da un gruppo di istanze. Per impostazione predefinita, un'istanza viene avviata senza alcun gruppo di sicurezza, il che significa che tutto il traffico IP in entrata, su qualsiasi porta TCP, verrà negato. Quindi, prima di inviare la nostra richiesta Spot, creeremo un gruppo di sicurezza che consenta il traffico di rete necessario. Ai fini di questo tutorial, creeremo un nuovo gruppo di sicurezza chiamato "GettingStarted" che consente il traffico Secure Shell (SSH) dall'indirizzo IP da cui viene eseguita l'applicazione. Per configurare un nuovo gruppo di sicurezza, è necessario includere o eseguire il seguente esempio di codice che configura il gruppo di sicurezza a livello di codice.

Dopo aver creato un oggetto `AmazonEC2` client, creiamo un `CreateSecurityGroupRequest` oggetto con il nome "GettingStarted" e una descrizione per il gruppo di sicurezza. Quindi chiamiamo l'`ec2.createSecurityGroup`API per creare il gruppo.

Per consentire l'accesso al gruppo, creiamo un `ipPermission` oggetto con l'intervallo di indirizzi IP impostato sulla rappresentazione CIDR della sottorete per il computer locale; il suffisso «/10" sull'indirizzo IP indica la sottorete per l'indirizzo IP specificato. Configuriamo l'`ipPermission`oggetto anche con il protocollo TCP e la porta 22 (SSH). Il passaggio finale consiste nella chiamata `ec2.authorizeSecurityGroupIngress` con il nome del nostro gruppo di sicurezza e dell'`ipPermission`oggetto.

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

Nota che è necessario eseguire questa applicazione una sola volta per creare un nuovo gruppo di sicurezza.

È inoltre possibile creare il gruppo di sicurezza utilizzando AWS Toolkit for Eclipse. Per ulteriori informazioni, vedere Gestione dei gruppi AWS Cost Explorer di sicurezza di.

Fase 3: Invio della richiesta Spot

Per inviare una richiesta Spot, devi prima determinare il tipo di istanza, Amazon Machine Image (AMI) e il prezzo di offerta massimo che desideri utilizzare. Devi anche includere il gruppo di sicurezza che abbiamo configurato in precedenza, in modo da poter accedere all'istanza, se lo desideri.

Esistono diversi tipi di istanze tra cui scegliere; vai a Tipi di Amazon EC2 istanze per un elenco completo. Per questo tutorial, useremo t1.micro, il tipo di istanza più economico disponibile. Successivamente, determineremo il tipo di AMI che vorremmo utilizzare. Useremo ami-a9d09ed1, l'AMI up-to-date Amazon Linux più disponibile quando abbiamo scritto questo tutorial. L'AMI più recente può cambiare nel tempo, ma puoi sempre determinare l'AMI della versione più recente seguendo questi passaggi:

1. Apri la Amazon EC2 console.
2. Scegli il pulsante Launch Instance.
3. La prima finestra mostra le AMIs opzioni disponibili. L'ID AMI è elencato accanto a ciascun titolo AMI. In alternativa, puoi utilizzare l'`DescribeImages`API, ma l'utilizzo di quel comando non rientra nell'ambito di questo tutorial.

Esistono molti modi per affrontare le offerte per le istanze Spot; per avere un'ampia panoramica dei vari approcci, ti consigliamo di guardare il video Bidding for Spot Instances. Tuttavia, per iniziare, descriveremo tre strategie comuni: fare un'offerta per garantire che il costo sia inferiore a quello dei prezzi su richiesta; fare un'offerta basata sul valore del calcolo risultante; presentare un'offerta in modo da acquisire capacità di calcolo il più rapidamente possibile.

- Riduci i costi al di sotto di quelli su richiesta Hai un processo di elaborazione in batch che richiederà diverse ore o giorni per essere eseguito. Tuttavia, sei flessibile rispetto a quando inizia e quando viene completato. Vuoi vedere se riesci a completarlo a un costo inferiore rispetto alle istanze on demand. Esamini la cronologia dei prezzi Spot per i tipi di istanza utilizzando l'API Console di gestione AWS o l' Amazon EC2 API. Per ulteriori informazioni, vedi Visualizzazione della cronologia dei prezzi Spot. Dopo aver analizzato la cronologia dei prezzi per il tipo di istanza desiderato in una determinata zona di disponibilità, hai due approcci alternativi per la tua offerta:
  - Puoi fare un'offerta nella fascia alta della gamma di prezzi Spot (ma comunque inferiore al prezzo on demand), prevedendo che la tua singola richiesta Spot sarà probabilmente soddisfatta ed eseguita per un tempo di elaborazione consecutivo sufficiente a completare il processo.
  - In alternativa, puoi specificare l'importo che sei disposto a pagare per le istanze Spot come percentuale del prezzo delle istanze on demand e pianificare di combinare molte istanze lanciate

nel tempo tramite una richiesta persistente. Se il prezzo specificato è superiore, l'istanza Spot viene terminata. (Più avanti nel tutorial spiegheremo come automatizzare questa operazione).

- Non pagate più del valore del risultato Avete un processo di elaborazione dati da eseguire. Conosci abbastanza il valore dei risultati del processo per sapere quando valgono in termini di costi di elaborazione. Dopo aver analizzato la cronologia dei prezzi Spot per il tipo di istanza, scegli un prezzo di offerta al quale il costo del tempo di elaborazione non sia superiore al valore dei risultati del lavoro. Crea un'offerta persistente e impostane l'esecuzione intermittente quando il prezzo Spot raggiunge o scende sotto la tua offerta.
- Acquisisci rapidamente la capacità di elaborazione Hai un bisogno imprevisto e a breve termine di capacità aggiuntiva che non è disponibile tramite le istanze on demand. Dopo aver analizzato la cronologia dei prezzi Spot per il tuo tipo di istanza, fai un'offerta superiore al prezzo storico più alto per avere un'alta probabilità che la tua richiesta venga soddisfatta rapidamente e che continui a essere elaborata fino al completamento.

Dopo aver scelto il prezzo di offerta, sei pronto a richiedere un'istanza Spot. Ai fini di questo tutorial, offriremo il prezzo su richiesta (0,03 USD) per massimizzare le possibilità che l'offerta venga soddisfatta. Puoi determinare i tipi di istanze disponibili e i prezzi su richiesta per le istanze accedendo alla pagina dei prezzi. Amazon EC2 Mentre un'istanza Spot è in esecuzione, paghi il prezzo Spot in vigore per il periodo di funzionamento delle istanze. I prezzi delle istanze Spot vengono fissati Amazon EC2 e adattati gradualmente in base alle tendenze a lungo termine della domanda e dell'offerta di capacità delle istanze Spot. Puoi anche specificare l'importo che sei disposto a pagare per un'istanza Spot come% del prezzo dell'istanza on demand. Per richiedere un'istanza Spot, devi semplicemente creare la tua richiesta con i parametri scelti in precedenza. Iniziamo con la creazione di un oggetto. `RequestSpotInstanceRequest` L'oggetto della richiesta richiede il numero di istanze che desideri avviare e il prezzo dell'offerta. Inoltre, è necessario impostare `LaunchSpecification` per la richiesta, che include il tipo di istanza, l'ID AMI e il gruppo di sicurezza che si desidera utilizzare. Una volta compilata la richiesta, chiamate il `requestSpotInstances` metodo sull'`AmazonEC2Client`oggetto. L'esempio seguente mostra come richiedere un'istanza Spot.

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

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

// Request 1 x t1.micro instance with a bid price of $0.03.
```

```
requestRequest.setSpotPrice("0.03");
requestRequest.setInstanceCount(Integer.valueOf(1));

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

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

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

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

L'esecuzione di questo codice avvierà una nuova richiesta di istanza Spot. Esistono altre opzioni che puoi utilizzare per configurare le tue richieste Spot. Per saperne di più, consulta Tutorial: Advanced Amazon EC2 Spot Request Management o la RequestSpotInstancesclasse nell' AWS SDK per Java API Reference.

> **Note**
>
> Ti verranno addebitati i costi per tutte le istanze Spot effettivamente lanciate, quindi assicurati di annullare tutte le richieste e di chiudere tutte le istanze avviate per ridurre le commissioni associate.

Fase 4: Determinare lo stato della richiesta Spot

Successivamente, vogliamo creare un codice per attendere che la richiesta Spot raggiunga lo stato «attivo» prima di procedere all'ultimo passaggio. Per determinare lo stato della nostra richiesta Spot, analizziamo il metodo describeSpotInstanceRequests in base allo stato dell'ID della richiesta Spot che vogliamo monitorare.

L'ID della richiesta creato nella Fase 2 è incorporato nella risposta alla nostra `requestSpotInstances` richiesta. Il codice di esempio seguente mostra come raccogliere le richieste IDs dalla `requestSpotInstances` risposta e utilizzarle per compilare un`ArrayList`.

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

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

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

Per monitorare l'ID della richiesta, chiamate il `describeSpotInstanceRequests` metodo per determinare lo stato della richiesta. Quindi ripeti finché la richiesta non si trova nello stato «aperto». Tieni presente che monitoriamo uno stato non «aperto», ma piuttosto uno stato, ad esempio, «attivo», perché la richiesta può passare direttamente a «chiusa» se c'è un problema con gli argomenti della richiesta. Il seguente esempio di codice fornisce i dettagli su come eseguire questa operazione.

```
// 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);
```

Dopo aver eseguito questo codice, la richiesta di istanza Spot sarà stata completata o avrà avuto esito negativo con un errore che verrà visualizzato sullo schermo. In entrambi i casi, possiamo procedere al passaggio successivo per ripulire eventuali richieste attive e terminare tutte le istanze in esecuzione.

Fase 5: Pulizia delle richieste e delle istanze Spot

Infine, dobbiamo ripulire le nostre richieste e istanze. È importante annullare eventuali richieste in sospeso e terminare eventuali istanze. La semplice cancellazione delle richieste non comporterà l'interruzione delle istanze, il che significa che continuerai a pagarle. Se chiudi le istanze, le tue

richieste Spot potrebbero essere annullate, ma in alcuni scenari, ad esempio nel caso in cui utilizzi offerte persistenti, la chiusura delle istanze non è sufficiente per impedire che la richiesta venga nuovamente soddisfatta. Pertanto, è consigliabile sia annullare le offerte attive che terminare le istanze in esecuzione.

Il codice seguente mostra come annullare le richieste.

```
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());
}
```

Per terminare le istanze in sospeso, è necessario l'ID dell'istanza associato alla richiesta che le ha avviate. Il seguente esempio di codice prende il nostro codice originale per il monitoraggio delle istanze e ne aggiunge uno `ArrayList` in cui memorizziamo l'ID dell'istanza associato alla risposta. `describeInstance`

```
// 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);
```

Utilizzando l'istanza IDs, memorizzata in`ArrayList`, terminate tutte le istanze in esecuzione utilizzando il seguente frammento di codice.

```
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());
}
```

Riunire tutto

Per mettere insieme tutto questo, forniamo un approccio più orientato agli oggetti che combina i passaggi precedenti che abbiamo mostrato: inizializzazione del EC2 client, invio della richiesta Spot, determinazione quando le richieste Spot non sono più nello stato aperto e pulizia di qualsiasi richiesta Spot persistente e delle istanze associate. Creiamo una classe chiamata che esegue queste azioni. `Requests`

Creiamo anche una `GettingStartedApp` classe, che ha un metodo principale in cui eseguiamo le chiamate di funzione di alto livello. In particolare, inizializziamo l'`Requests`oggetto descritto in precedenza. Inoltriamo la richiesta di istanza Spot. Quindi aspettiamo che la richiesta Spot raggiunga lo stato «Attivo». Infine, puliamo le richieste e le istanze.

Il codice sorgente completo di questo esempio può essere visualizzato o scaricato all'indirizzo GitHub.

Complimenti! Hai appena completato il tutorial introduttivo per lo sviluppo del software Spot Instance con AWS SDK per Java.

Fasi successive

Procedi con il tutorial: Advanced Amazon EC2 Spot Request Management.

Tutorial: Gestione avanzata delle richieste Amazon EC2 Spot

Amazon EC2 Le istanze Spot ti consentono di fare offerte sulla Amazon EC2 capacità inutilizzata e di eseguire tali istanze finché l'offerta supera il prezzo spot corrente. Amazon EC2 modifica periodicamente il prezzo spot in base alla domanda e all'offerta. Per ulteriori informazioni sulle istanze Spot, consulta le istanze Spot nella Guida per l' Amazon EC2 utente delle istanze Linux.

Prerequisiti

Per utilizzare questo tutorial è necessario averle AWS SDK per Java installate, oltre a soddisfare i relativi prerequisiti di installazione di base. Per ulteriori informazioni, consulta Configurare il AWS SDK per Java.

Configurazione delle credenziali

Per iniziare a utilizzare questo esempio di codice, è necessario impostare AWS le credenziali. Per istruzioni su come eseguire questa operazione, consulta Configurare le AWS credenziali e la regione per lo sviluppo.

> **Note**
>
> Ti consigliamo di utilizzare le credenziali di un IAM utente per fornire questi valori. Per ulteriori informazioni, consulta Registrazione AWS e creazione di un IAM utente.

Dopo aver configurato le impostazioni, puoi iniziare a utilizzare il codice riportato nell'esempio.

Configurazione di un gruppo di sicurezza

Un gruppo di sicurezza funge da firewall che controlla il traffico consentito in entrata e in uscita da un gruppo di istanze. Per impostazione predefinita, un'istanza viene avviata senza alcun gruppo di sicurezza, il che significa che tutto il traffico IP in entrata, su qualsiasi porta TCP, verrà negato. Quindi, prima di inviare la nostra richiesta Spot, creeremo un gruppo di sicurezza che consenta il traffico di rete necessario. Ai fini di questo tutorial, creeremo un nuovo gruppo di sicurezza chiamato "GettingStarted" che consente il traffico Secure Shell (SSH) dall'indirizzo IP da cui viene eseguita l'applicazione. Per configurare un nuovo gruppo di sicurezza, è necessario includere o eseguire il seguente esempio di codice che configura il gruppo di sicurezza a livello di codice.

Dopo aver creato un oggetto `AmazonEC2` client, creiamo un `CreateSecurityGroupRequest` oggetto con il nome "GettingStarted" e una descrizione per il gruppo di sicurezza. Quindi chiamiamo l'`ec2.createSecurityGroup`API per creare il gruppo.

Per consentire l'accesso al gruppo, creiamo un `ipPermission` oggetto con l'intervallo di indirizzi IP impostato sulla rappresentazione CIDR della sottorete per il computer locale; il suffisso «/10" sull'indirizzo IP indica la sottorete per l'indirizzo IP specificato. Configuriamo l'`ipPermission`oggetto anche con il protocollo TCP e la porta 22 (SSH). Il passaggio finale consiste nella chiamata

`ec2 .authorizeSecurityGroupIngress` con il nome del nostro gruppo di sicurezza e dell'`ipPermission`oggetto.

(Il codice seguente è lo stesso che abbiamo usato nel primo tutorial.)

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

È possibile visualizzare l'intero esempio di codice nell'esempio di `advanced.CreateSecurityGroupApp.java` codice. Nota: è necessario eseguire questa applicazione una sola volta per creare un nuovo gruppo di sicurezza.

> **Note**
>
> È inoltre possibile creare il gruppo di sicurezza utilizzando AWS Toolkit for Eclipse. Per ulteriori informazioni, vedere Gestione dei gruppi di sicurezza AWS Cost Explorer nella Guida per AWS Toolkit for Eclipse l'utente.

Opzioni dettagliate per la creazione di richieste di istanze Spot

Come spiegato in Tutorial: Amazon EC2 Spot Instances, devi creare la tua richiesta con un tipo di istanza, un'Amazon Machine Image (AMI) e un prezzo di offerta massimo.

Cominciamo con la creazione di un `RequestSpotInstanceRequest` oggetto. L'oggetto della richiesta richiede il numero di istanze desiderate e il prezzo dell'offerta. Inoltre, dobbiamo impostare `LaunchSpecification` per la richiesta, che include il tipo di istanza, l'ID AMI e il gruppo di sicurezza che desideri utilizzare. Dopo che la richiesta è stata compilata, chiamiamo il `requestSpotInstances` metodo sull'`AmazonEC2Client`oggetto. Segue un esempio di come richiedere un'istanza Spot.

(Il codice seguente è lo stesso che abbiamo usato nel primo tutorial.)

```
// 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);
```

Richieste persistenti o richieste una tantum

Quando si crea una richiesta Spot, è possibile specificare diversi parametri opzionali. Il primo è se la richiesta è una tantum o persistente. Per impostazione predefinita, è una richiesta una tantum. Una richiesta unica può essere soddisfatta una sola volta e, una volta terminate le istanze richieste, la richiesta verrà chiusa. Una richiesta persistente viene presa in considerazione ai fini dell'evasione ogni volta che non è in esecuzione alcuna istanza Spot per la stessa richiesta. Per specificare il tipo di richiesta, è sufficiente impostare il Tipo nella richiesta Spot. Questo può essere fatto con il seguente codice.

```
// 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);
```

Limitazione della durata di una richiesta

Puoi anche specificare facoltativamente per quanto tempo la tua richiesta rimarrà valida. È possibile specificare sia l'ora di inizio che quella di fine per questo periodo. Per impostazione predefinita, una richiesta Spot viene presa in considerazione per l'evasione dal momento in cui viene creata fino a quando non viene soddisfatta o annullata da te. Tuttavia, puoi limitare il periodo di validità, se necessario. Un esempio di come specificare questo periodo è mostrato nel codice seguente.

```
// 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);
```

Raggruppamento delle richieste di istanze Amazon EC2 Spot

Hai la possibilità di raggruppare le richieste di istanze Spot in diversi modi. Esamineremo i vantaggi dell'utilizzo di gruppi di lancio, gruppi di zone di disponibilità e gruppi di collocamento.

Se vuoi assicurarti che le tue istanze Spot vengano lanciate e chiuse tutte insieme, hai la possibilità di avvalerti di un gruppo di lancio. Un gruppo di lancio è un'etichetta che raggruppa una serie di offerte. Tutte le istanze in un gruppo di avvio vengono avviate e terminate insieme. Nota: se le istanze di un gruppo di lancio sono già state soddisfatte, non vi è alcuna garanzia che vengano soddisfatte anche le nuove istanze lanciate con lo stesso gruppo di lancio. Un esempio di come impostare un Launch Group è mostrato nel seguente esempio di codice.

```
// 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);
```

Se vuoi assicurarti che tutte le istanze all'interno di una richiesta vengano lanciate nella stessa zona di disponibilità e non ti interessa quale, puoi sfruttare i gruppi di zone di disponibilità. Un gruppo di zone di disponibilità è un'etichetta che raggruppa un insieme di istanze nella stessa zona di disponibilità. Tutte le istanze che condividono un gruppo di zone di disponibilità e vengono gestite contemporaneamente inizieranno nella stessa zona di disponibilità. Segue un esempio di come impostare un gruppo di zone di disponibilità.

```
// 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);
```

Puoi specificare una zona di disponibilità che desideri per le tue istanze Spot. Il seguente esempio di codice mostra come impostare una zona di disponibilità.

```
// 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);
```

Infine, è possibile specificare un gruppo di collocamento se si utilizzano istanze Spot HPC (High Performance Computing), come istanze di calcolo in cluster o istanze GPU in cluster. I gruppi di posizionamento offrono una latenza inferiore e una connettività a larghezza di banda elevata tra le istanze. Segue un esempio di come impostare un gruppo di collocamento.

```
// 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);
```

Tutti i parametri mostrati in questa sezione sono opzionali. È anche importante rendersi conto che la maggior parte di questi parametri, ad eccezione del fatto che l'offerta sia una tantum o permanente, può ridurre la probabilità che l'offerta venga soddisfatta. Pertanto, è importante sfruttare queste opzioni solo se ne hai bisogno. Tutti gli esempi di codice precedenti sono combinati in un unico lungo esempio di codice, che può essere trovato nella `com.amazonaws.codesamples.advanced.InlineGettingStartedCodeSampleApp.java` classe.

Come rendere persistente una partizione root dopo l'interruzione o la terminazione

Uno dei modi più semplici per gestire l'interruzione delle istanze Spot consiste nell'assicurare che i dati vengano indirizzati a un volume Amazon Elastic Block Store (Amazon Amazon EBS) con cadenza regolare. Effettuando il checkpoint periodicamente, in caso di interruzione perderai solo i dati creati dall'ultimo checkpoint (supponendo che nel frattempo non vengano eseguite altre azioni non idempotenti). Per semplificare questo processo, puoi configurare la tua Spot Request in modo che la partizione root non venga eliminata in caso di interruzione o chiusura. Nell'esempio seguente abbiamo inserito un nuovo codice che mostra come abilitare questo scenario.

Nel codice aggiunto, creiamo un `BlockDeviceMapping` oggetto e impostiamo il relativo oggetto associato Amazon Elastic Block Store (Amazon EBS) su un Amazon EBS oggetto che abbiamo configurato per `not` essere eliminato se l'istanza Spot viene terminata. Lo aggiungiamo quindi `BlockDeviceMapping` alle ArrayList mappature che includiamo nelle specifiche di lancio.

```
// 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);
```

Supponendo che tu voglia ricollegare questo volume alla tua istanza all'avvio, puoi anche utilizzare le impostazioni di mappatura dei dispositivi a blocchi. In alternativa, se hai collegato una partizione non root, puoi specificare i Amazon EBS volumi Amazon che desideri collegare all'istanza Spot dopo la ripresa. Puoi farlo semplicemente specificando un ID di istantanea nel tuo oggetto `EbsBlockDevice` e un nome di dispositivo alternativo negli oggetti. `BlockDeviceMapping` Sfruttando le mappature dei dispositivi a blocchi, può essere più semplice avviare l'istanza.

L'utilizzo della partizione root per il checkpoint dei dati critici è un ottimo modo per gestire il potenziale di interruzione delle istanze. Per ulteriori metodi di gestione del potenziale di interruzione, guarda il video sulla gestione delle interruzioni.

Come etichettare le richieste e le istanze spot

L'aggiunta di tag alle Amazon EC2 risorse può semplificare l'amministrazione dell'infrastruttura cloud. I tag, una forma di metadati, possono essere utilizzati per creare nomi intuitivi, migliorare la ricercabilità e migliorare il coordinamento tra più utenti. Puoi anche utilizzare i tag per automatizzare script e parti dei tuoi processi. Per saperne di più sull'etichettatura Amazon EC2 delle risorse, consulta la sezione Uso dei tag nella Guida Amazon EC2 utente per le istanze Linux.

Tagging delle richieste

Per aggiungere tag alle tue richieste Spot, devi taggarle dopo che sono state richieste. Il valore restituito da ti `requestSpotInstances()` fornisce un RequestSpotInstancesResultoggetto che puoi usare per ottenere la richiesta spot IDs per il tagging:

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

Una volta ottenuto il IDs, puoi taggare le richieste aggiungendole IDs a CreateTagsRequeste chiamando il `createTags()` metodo del Amazon EC2 client:

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

Taggare le istanze

Analogamente alle richieste spot stesse, puoi taggare un'istanza solo dopo averla creata, cosa che avverrà una volta soddisfatta la richiesta spot (non è più nello stato aperto).

Puoi controllare lo stato delle tue richieste chiamando il `describeSpotInstanceRequests()` metodo del Amazon EC2 client con un DescribeSpotInstanceRequestsRequestoggetto. L'DescribeSpotInstanceRequestsResultoggetto restituito contiene un elenco di SpotInstanceRequestoggetti che è possibile utilizzare per interrogare lo stato delle richieste Spot e ottenerne l'istanza IDs una volta che non sono più nello stato aperto.

Una volta che la richiesta spot non è più aperta, puoi recuperarne l'ID di istanza dall'`SpotInstanceRequest`oggetto chiamandone il `getInstanceId()` metodo.

```
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);
```

Ora puoi taggare le istanze restituite:

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

Annullamento delle richieste spot e chiusura delle istanze

Annullamento di una richiesta spot

Per annullare una richiesta di istanza Spot, chiama `cancelSpotInstanceRequests` il Amazon EC2 client con un CancelSpotInstanceRequestsRequestoggetto.

```
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());
}
```

Chiusura delle istanze Spot

Puoi terminare qualsiasi istanza Spot in esecuzione passandola IDs al metodo del Amazon EC2 client. `terminateInstances()`

```
try {
    TerminateInstancesRequest terminateRequest = new
 TerminateInstancesRequest(instanceIds);
    ec2.terminateInstances(terminateRequest);
} catch (AmazonServiceException e) {
    System.out.println("Error terminating instances");
    System.out.println("Caught Exception: " + e.getMessage());
    System.out.println("Reponse Status Code: " + e.getStatusCode());
    System.out.println("Error Code: " + e.getErrorCode());
    System.out.println("Request ID: " + e.getRequestId());
}
```

Riunire tutto

Per riunire tutto questo, forniamo un approccio più orientato agli oggetti che combina i passaggi mostrati in questo tutorial in un'unica classe facile da usare. Creiamo un'istanza di una classe chiamata `Requests` che esegue queste azioni. Creiamo anche una `GettingStartedApp` classe, che ha un metodo principale in cui eseguiamo le chiamate di funzione di alto livello.

Il codice sorgente completo di questo esempio può essere visualizzato o scaricato all'indirizzo GitHub.

Complimenti! Hai completato il tutorial Advanced Request Features per lo sviluppo del software Spot Instance con AWS SDK per Java.

Gestione delle Amazon EC2 istanze

Creazione di un'istanza

Crea una nuova Amazon EC2 istanza chiamando il `runInstances` metodo del EC2 client Amazon, fornendogli una Amazon Machine Image (AMI) RunInstancesRequestcontenente l'Amazon Machine Image (AMI) da utilizzare e un tipo di istanza.

Importazioni

```
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;
```

Codice

```
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();
```

Guarda l'esempio completo.

Avvio di un'istanza

Per avviare un' Amazon EC2 istanza, chiama il `startInstances` metodo del EC2 client Amazon, fornendogli un codice StartInstancesRequestcontenente l'ID dell'istanza da avviare.

Importazioni

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

Codice

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

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

ec2.startInstances(request);
```

Guarda l'esempio completo.

Arresto di un'istanza

Per interrompere un' Amazon EC2 istanza, chiama il `stopInstances` metodo del EC2 client Amazon, fornendogli un codice StopInstancesRequestcontenente l'ID dell'istanza da interrompere.

Importazioni

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

Codice

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

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

ec2.stopInstances(request);
```

Guarda l'esempio completo.

Riavvio di un'istanza

Per riavviare un' Amazon EC2 istanza, chiama il `rebootInstances` metodo del EC2 client Amazon, fornendogli un file RebootInstancesRequestcontenente l'ID dell'istanza da riavviare.

Importazioni

```
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;
```

Codice

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

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

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

Guarda l'esempio completo.

Descrizione di istanze

Per elencare le tue istanze, crea un metodo DescribeInstancesRequeste richiama il `describeInstances` metodo del EC2 client Amazon. Restituirà un DescribeInstancesResultoggetto che potrai utilizzare per elencare Amazon EC2 le istanze del tuo account e della tua regione.

Le istanze sono raggruppate in base alla prenotazione. Ogni prenotazione corrisponde alla chiamata a `startInstances` che ha avviato l'istanza. Per elencare le tue istanze, devi prima chiamare la `DescribeInstancesResult` 'classe' `getReservations' method, and then call` `getInstances su ogni oggetto Reservation restituito.

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

I risultati vengono visualizzati in una pagina; è possibile ottenere ulteriori risultati passando il valore restituito dal `getNextToken` metodo dell'oggetto risultato al `setNextToken` metodo dell'oggetto di richiesta originale, quindi utilizzando lo stesso oggetto di richiesta nella chiamata successiva a. `describeInstances`

Guarda l'esempio completo.

Monitoraggio di un'istanza

È possibile monitorare vari aspetti delle Amazon EC2 istanze, come l'utilizzo della CPU e della rete, la memoria disponibile e lo spazio rimanente su disco. Per ulteriori informazioni sul monitoraggio delle istanze, consulta Monitoraggio Amazon EC2 nella Guida per l' Amazon EC2 utente delle istanze Linux.

Per iniziare a monitorare un'istanza, devi crearne un'MonitorInstancesRequestcon l'ID dell'istanza da monitorare e passarla al `monitorInstances` metodo del EC2 client Amazon.

Importazioni

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

Codice

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

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

```
ec2.monitorInstances(request);
```

Guarda l'esempio completo.

Arresto del monitoraggio delle istanze

Per interrompere il monitoraggio di un'istanza, UnmonitorInstancesRequestcreane un'istanza con l'ID dell'istanza per interrompere il monitoraggio e passala al `unmonitorInstances` metodo del EC2 client Amazon.

Importazioni

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

Codice

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

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

ec2.unmonitorInstances(request);
```

Guarda l'esempio completo.

Ulteriori informazioni

- RunInstancesnell' Amazon EC2 API Reference
- DescribeInstancesnell' Amazon EC2 API Reference
- StartInstancesnell' Amazon EC2 API Reference
- StopInstancesnell' Amazon EC2 API Reference
- RebootInstancesnell' Amazon EC2 API Reference
- MonitorInstancesnell' Amazon EC2 API Reference
- UnmonitorInstancesnell' Amazon EC2 API Reference

Utilizzo di indirizzi IP elastici in Amazon EC2

EC2-Classic sta per andare in pensione

> **Warning**
>
> Ritireremo EC2 -Classic il 15 agosto 2022. Ti consigliamo di migrare da EC2 -Classic a un VPC. Per ulteriori informazioni, consulta il post sul blog EC2-Classic-Classic Networking is Retiring — Ecco come prepararsi.

Allocazione di un indirizzo IP elastico

Per utilizzare un indirizzo IP elastico bisogna prima allocarne uno al proprio account e associarlo con la propria istanza o con un'interfaccia di rete.

Per allocare un indirizzo IP elastico, chiama il `allocateAddress` metodo del EC2 client Amazon con un AllocateAddressRequestoggetto contenente il tipo di rete (classica EC2 o VPC).

Il file restituito AllocateAddressResultcontiene un ID di allocazione che puoi utilizzare per associare l'indirizzo a un'istanza, passando l'ID di allocazione e l'ID dell'istanza in a AssociateAddressRequestal metodo del EC2 `associateAddress` client Amazon.

Importazioni

```
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;
```

Codice

```
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);
```

Guarda l'esempio completo.

Descrizione degli indirizzi IP elastici

Per elencare gli indirizzi IP elastici assegnati al tuo account, chiama il `describeAddresses` metodo di Amazon EC2 Client. Restituisce un valore DescribeAddressesResultche puoi utilizzare per ottenere un elenco di oggetti Address che rappresentano gli indirizzi IP elastici del tuo account.

Importazioni

```
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;
```

Codice

```
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());
}
```

Guarda l'esempio completo.

Rilascio di un indirizzo IP elastico

Per rilasciare un indirizzo IP elastico, chiama il `releaseAddress` metodo del EC2 client Amazon, passandogli un indirizzo ReleaseAddressRequestcontenente l'ID di allocazione dell'indirizzo IP elastico che desideri rilasciare.

Importazioni

```
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;
```

Codice

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

ReleaseAddressRequest request = new ReleaseAddressRequest()
    .withAllocationId(alloc_id);

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

Dopo aver rilasciato un indirizzo IP elastico, questo viene rilasciato al pool di indirizzi AWS IP e in seguito potrebbe non essere più disponibile. Assicurati di aggiornare i record DNS e gli eventuali server o dispositivi che comunicano con l'indirizzo. Se tenti di rilasciare un indirizzo IP elastico che hai già rilasciato, riceverai un AuthFailureerrore se l'indirizzo è già assegnato a un altro. Account AWS

Se utilizzi EC2-Classic o un VPC predefinito, il rilascio di un indirizzo IP elastico lo dissocia automaticamente da qualsiasi istanza a cui è associato. Per dissociare un indirizzo IP elastico senza rilasciarlo, utilizza il metodo di `disassociateAddress` Amazon EC2 Client.

Se utilizzi un VPC non di default, devi utilizzare `disassociateAddress` per disassociare l'indirizzo IP elastico prima di provare a rilasciarlo. Altrimenti, Amazon EC2 restituisce un errore (non valido. IPAddress InUse).

Vedi l'esempio completo.

Ulteriori informazioni

- Indirizzi IP elastici nella Guida per l' Amazon EC2 utente per le istanze Linux
- AllocateAddressnell' Amazon EC2 API Reference
- DescribeAddressesnell' Amazon EC2 API Reference
- ReleaseAddressnell' Amazon EC2 API Reference

Usa aree e zone di disponibilità

Descrivere le regioni

Per elencare le regioni disponibili per il tuo account, chiama il `describeRegions` metodo del EC2 cliente Amazon. Restituisce DescribeRegionsResult. Chiamare il metodo `getRegions` dell'oggetto restituito per ottenere un elenco di oggetti Region che rappresentano ciascuna regione.

Importazioni

```
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;
```

Codice

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

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

Guarda l'esempio completo.

Descrivere le zone di disponibilità

Per elencare ogni zona di disponibilità disponibile per il tuo account, chiama il `describeAvailabilityZones` metodo del EC2 client Amazon. Restituisce DescribeAvailabilityZonesResult. Chiama il suo `getAvailabilityZones` metodo per ottenere un elenco di AvailabilityZoneoggetti che rappresentano ogni zona di disponibilità.

Importazioni

```
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;
```

Codice

```
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());
}
```

Guarda l'esempio completo.

Descrivere gli account

Per descrivere il tuo account, chiama il `describeAccountAttributes` metodo del EC2 client Amazon. Questo metodo restituisce un DescribeAccountAttributesResultoggetto. Invoca questo `getAccountAttributes` metodo degli oggetti per ottenere un elenco di AccountAttributeoggetti. È possibile scorrere l'elenco per recuperare un oggetto. AccountAttribute

Puoi ottenere i valori degli attributi del tuo account richiamando il metodo dell'AccountAttributeoggetto. `getAttributeValues` Questo metodo restituisce un elenco di

AccountAttributeValueoggetti. È possibile scorrere questo secondo elenco per visualizzare il valore degli attributi (vedere l'esempio di codice riportato di seguito).

Importazioni

```
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;
```

Codice

```
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();
}
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Regioni e zone di disponibilità nella Guida Amazon EC2 utente per le istanze Linux
- DescribeRegionsnel riferimento Amazon EC2 API
- DescribeAvailabilityZonesnell' Amazon EC2 API Reference

Lavorare con coppie di Amazon EC2 chiavi

Creazione di una coppia di chiavi

Per creare una coppia di chiavi, chiama il `createKeyPair` metodo del EC2 client Amazon con un CreateKeyPairRequestche contiene il nome della chiave.

Importazioni

```
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;
```

Codice

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

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

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

Guarda l'esempio completo.

Descrizione delle coppie di chiavi

Per elencare le tue coppie di chiavi o per ottenere informazioni su di esse, chiama il `describeKeyPairs` metodo del EC2 client Amazon. Restituisce un DescribeKeyPairsResultcomando che puoi usare per accedere all'elenco delle coppie di chiavi chiamando il relativo `getKeyPairs` metodo, che restituisce un elenco di KeyPairInfooggetti.

Importazioni

```
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;
```

Codice

```
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());
}
```

Vedi l'esempio completo.

Eliminazione di una coppia di chiavi

Per eliminare una coppia di chiavi, chiama il `deleteKeyPair` metodo del EC2 client Amazon, passandogli un codice DeleteKeyPairRequestcontenente il nome della coppia di chiavi da eliminare.

Importazioni

```
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;
```

Codice

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

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

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

Guarda l'esempio completo.

Ulteriori informazioni

- Amazon EC2 Coppie di chiavi nella Guida per l' Amazon EC2 utente per le istanze Linux
- CreateKeyPairnel riferimento alle Amazon EC2 API
- DescribeKeyPairsnell' Amazon EC2 API Reference
- DeleteKeyPairnell' Amazon EC2 API Reference

Lavorare con i gruppi di sicurezza in Amazon EC2

Creazione di un gruppo di sicurezza

Per creare un gruppo di sicurezza, chiama il `createSecurityGroup` metodo del EC2 client Amazon con un CreateSecurityGroupRequestche contiene il nome della chiave.

Importazioni

```
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;
```

Codice

```
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);
```

Guarda l'esempio completo.

Configurazione di un gruppo di sicurezza

Un gruppo di sicurezza può controllare sia il traffico in entrata (ingresso) che quello in uscita (in uscita) verso le tue istanze. Amazon EC2

Per aggiungere regole di ingresso al tuo gruppo di sicurezza, usa il `authorizeSecurityGroupIngress` metodo di Amazon EC2 Client, fornendo il nome del gruppo di sicurezza e le regole di accesso (IpPermission) che desideri assegnargli all'interno di un AuthorizeSecurityGroupIngressRequestoggetto. Nell'esempio seguente viene mostrato come aggiungere autorizzazioni IP a un gruppo di sicurezza.

Importazioni

```
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;
```

Codice

```
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);
```

Per aggiungere una regola di uscita al gruppo di sicurezza, fornisci dati simili nel `authorizeSecurityGroupEgress` metodo del EC2 cliente Amazon. AuthorizeSecurityGroupEgressRequest

Guarda l'esempio completo.

Descrizione di gruppi di sicurezza

Per descrivere i tuoi gruppi di sicurezza o ottenere informazioni su di essi, chiama il `describeSecurityGroups` metodo di Amazon EC2 Client. Restituisce un DescribeSecurityGroupsResultcomando che puoi usare per accedere all'elenco dei gruppi di sicurezza chiamando il relativo `getSecurityGroups` metodo, che restituisce un elenco di SecurityGroupoggetti.

Importazioni

```
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;
```

Codice

```
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];
```

Vedi l'esempio completo.

Eliminazione di un gruppo di sicurezza

Per eliminare un gruppo di sicurezza, chiama il `deleteSecurityGroup` metodo del EC2 client Amazon, passandogli un gruppo di sicurezza DeleteSecurityGroupRequestche contiene l'ID del gruppo di sicurezza da eliminare.

Importazioni

```
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;
```

Codice

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

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

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

Guarda l'esempio completo.

Ulteriori informazioni

- Amazon EC2 Gruppi di sicurezza nella Guida per l' Amazon EC2 utente per le istanze Linux
- Autorizzazione del traffico in entrata per le istanze Linux nella Guida per l' Amazon EC2 utente per le istanze Linux
- CreateSecurityGroupnella guida di riferimento alle API Amazon EC2
- DescribeSecurityGroupsnell' Amazon EC2 API Reference
- DeleteSecurityGroupnell' Amazon EC2 API Reference
- AuthorizeSecurityGroupIngressnell' Amazon EC2 API Reference

Esempi IAM che utilizzano il AWS SDK per Java

In questa sezione vengono forniti esempi di programmazione di IAM con AWS SDK per Java.

AWS Identity and Access Management (IAM) consente di controllare in modo sicuro l'accesso ai AWS servizi e alle risorse per gli utenti. Utilizzando IAM, puoi creare e gestire AWS utenti e gruppi e utilizzare le autorizzazioni per consentire e negare il loro accesso alle risorse. AWS Per una guida completa a IAM, consulta la Guida per l'IAM utente.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ogni tecnica. Il codice di esempio completo è disponibile su GitHub. Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

Argomenti

Gestione delle chiavi di accesso IAM

Creazione di una chiave di accesso

Per creare una chiave di accesso IAM, chiama il AmazonIdentityManagementClient `createAccessKey` metodo con un CreateAccessKeyRequestoggetto.

`CreateAccessKeyRequest`ha due costruttori, uno che richiede un nome utente e l'altro senza parametri. Se utilizzate la versione che non accetta parametri, dovete impostare il nome utente utilizzando il metodo `withUserName` setter prima di passarlo al `createAccessKey` metodo.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su. GitHub

Elencazione delle chiavi di accesso

Per elencare le chiavi di accesso per un determinato utente, create un ListAccessKeysRequestoggetto che contenga il nome utente per cui elencare le chiavi e passatelo al `listAccessKeys` metodo AmazonIdentityManagementClient's.

> **Note**
>
> Se non fornite un nome utente`listAccessKeys`, tenterà di elencare le chiavi di accesso associate a chi Account AWS ha firmato la richiesta.

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

I risultati di `listAccessKeys` sono paginati (con un massimo predefinito di 100 record per chiamata). È possibile richiamare `getIsTruncated` l'ListAccessKeysResultoggetto restituito per verificare se la query ha restituito un numero inferiore di risultati rispetto a quelli disponibili. In tal caso, chiamate `setMarker` `ListAccessKeysRequest` e passatelo alla successiva invocazione di. `listAccessKeys`

Vedi l'esempio completo su. GitHub

Recupero dell'ora ultimo utilizzo di una chiave di accesso

Per sapere l'ora in cui è stata utilizzata l'ultima volta una chiave di accesso, chiamate il `getAccessKeyLastUsed` metodo AmazonIdentityManagementClient's con l'ID della chiave di accesso (che può essere passato utilizzando un GetAccessKeyLastUsedRequestoggetto) o direttamente all'overload che accetta direttamente l'ID della chiave di accesso.

È quindi possibile utilizzare l'GetAccessKeyLastUsedResultoggetto restituito per recuperare l'ultima ora utilizzata della chiave.

Importazioni

```
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;
```

Codice

```
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());
```

Vedi l'esempio completo su. GitHub

Attivazione o disattivazione delle chiavi di accesso

È possibile attivare o disattivare una chiave di accesso creando un UpdateAccessKeyRequestoggetto, fornendo l'ID della chiave di accesso, facoltativamente il nome utente e lo Status desiderato, quindi passando l'oggetto della richiesta al metodo AmazonIdentityManagementClient's`updateAccessKey`.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su. GitHub

Eliminazione di una chiave di accesso

Per eliminare definitivamente una chiave di accesso, chiama il `deleteKey` metodo AmazonIdentityManagementClient's, fornendogli un ID e il nome utente DeleteAccessKeyRequestcontenenti l'ID e il nome utente della chiave di accesso.

> **Note**
>
> Dopo che è stata eliminata, una chiave non può più essere recuperata né utilizzata. Per disattivare temporaneamente una chiave in modo che possa essere riattivata in un secondo momento, usa invece updateAccessKeymethod.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su. GitHub

Ulteriori informazioni

- CreateAccessKeynel riferimento all'API IAM
- ListAccessKeysnel riferimento all'API IAM
- GetAccessKeyLastUsednel riferimento all'API IAM
- UpdateAccessKeynel riferimento all'API IAM

- DeleteAccessKeynel riferimento all'API IAM

Gestione degli utenti IAM

Creazione di un utente

Crea un nuovo utente IAM fornendo il nome utente al `createUser` metodo AmazonIdentityManagementClient's, direttamente o utilizzando un CreateUserRequestoggetto contenente il nome utente.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su GitHub.

Elencazione di utenti

Per elencare gli utenti IAM del tuo account, creane uno nuovo ListUsersRequeste passalo al `listUsers` metodo AmazonIdentityManagementClient's. Puoi recuperare l'elenco degli utenti `getUsers` chiamando l'ListUsersResultoggetto restituito.

L'elenco di utenti restituito da `listUsers` è paginato. È possibile verificare se ci sono più risultati da recuperare chiamando il metodo `getIsTruncated` dell'oggetto di risposta. Se restituisce`true`, chiama il `setMarker()` metodo dell'oggetto di richiesta, passandogli il valore restituito dal `getMarker()` metodo dell'oggetto di risposta.

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

Vedi l'esempio completo su GitHub.

Aggiornamento di un utente

Per aggiornare un utente, chiamate il `updateUser` metodo dell' AmazonIdentityManagementClient oggetto, che accetta un UpdateUserRequestoggetto che potete usare per modificare il nome o il percorso dell'utente.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su GitHub.

Eliminazione di un utente

Per eliminare un utente, chiama la `deleteUser` richiesta AmazonIdentityManagementClient's con un UpdateUserRequestoggetto impostato con il nome utente da eliminare.

Importazioni

```
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;
```

Codice

```
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;
```

```
}
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Utenti IAM nella Guida IAM per l'utente
- Gestione degli utenti IAM nella Guida IAM per l'utente
- CreateUsernella guida di riferimento all'API IAM
- ListUsersnel riferimento all'API IAM
- UpdateUsernel riferimento all'API IAM
- DeleteUsernel riferimento all'API IAM

Utilizzo di alias dell'account IAM

Se desideri che l'URL della tua pagina di accesso contenga il nome della tua azienda o un altro identificativo descrittivo anziché il tuo Account AWS ID, puoi creare un alias per il tuo. Account AWS

> **Note**
>
> AWS supporta esattamente un alias di account per account.

Creazione di un alias dell'account

Per creare un alias di account, chiama il `createAccountAlias` metodo AmazonIdentityManagementClient's con un CreateAccountAliasRequestoggetto che contiene il nome dell'alias.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su. GitHub

Elencazione di alias dell'account

Per elencare l'alias del tuo account, se esiste, chiama il `listAccountAliases` metodo AmazonIdentityManagementClient's.

> **Note**
>
> Il metodo restituito ListAccountAliasesResultsupporta gli stessi `getMarker` metodi `getIsTruncated` e AWS SDK per Java degli altri metodi di elenco, ma Account AWS può avere un solo alias di account.

importazioni

```
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);
}
```

vedi l'esempio completo su. GitHub

Eliminazione di un alias dell'account

Per eliminare l'alias del tuo account, chiama il `deleteAccountAlias` metodo AmazonIdentityManagementClient's. Quando si elimina l'alias di un account, è necessario fornirne il nome utilizzando un oggetto. DeleteAccountAliasRequest

importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su. GitHub

Ulteriori informazioni

- L'ID AWS dell'account e il relativo alias nella Guida per l' IAM utente
- CreateAccountAliasnel riferimento all'API IAM
- ListAccountAliasesnel riferimento all'API IAM
- DeleteAccountAliasnel riferimento all'API IAM

Lavorare con le policy IAM

Creazione di una policy

Per creare una nuova policy, fornisci il nome della policy e un documento di policy in formato JSON utilizzando un CreatePolicyRequestmetodo to the's. AmazonIdentityManagementClient `createPolicy`

Importazioni

```
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;
```

Codice

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

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

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

I documenti relativi alle policy IAM sono stringhe JSON con una sintassi ben documentata.
Nell'esempio che segue viene fornito l'accesso per effettuare richieste particolari a DynamoDB.

```
public static final String POLICY_DOCUMENT =
    "{" +
    "  \"Version\": \"2012-10-17\",       " +
    "  \"Statement\": [" +
    "    {" +
    "        \"Effect\": \"Allow\"," +
    "        \"Action\": \"logs:CreateLogGroup\"," +
    "        \"Resource\": \"%s\"" +
    "    }," +
    "    {" +
    "        \"Effect\": \"Allow\"," +
    "        \"Action\": [" +
    "            \"dynamodb:DeleteItem\"," +
    "            \"dynamodb:GetItem\"," +
    "            \"dynamodb:PutItem\"," +
    "            \"dynamodb:Scan\"," +
    "            \"dynamodb:UpdateItem\"" +
    "       ]," +
    "       \"Resource\": \"RESOURCE_ARN\"" +
    "    }" +
    "   ]" +
```

```
    "}";
```

Vedi l'esempio completo su. GitHub

Recupero di una policy

Per recuperare una policy esistente, chiamate il `getPolicy` metodo AmazonIdentityManagementClient's, fornendo l'ARN della policy all'interno GetPolicyRequestdi un oggetto.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su. GitHub

Collegamento di una policy del ruolo

Puoi allegare una policy a IAMhttp: //docs.aws.amazon. com/IAM/latest/UserGuide/id_roles.html [role] chiamando il `attachRolePolicy` metodo AmazonIdentityManagementClient's, fornendogli il nome del ruolo e l'ARN della policy in un. AttachRolePolicyRequest

Importazioni

```
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;
```

Codice

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

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

iam.attachRolePolicy(attach_request);
```

Vedi l'esempio completo su. GitHub

Elencazione di policy dei ruoli collegate

Elenca le politiche allegate su un ruolo chiamando il `listAttachedRolePolicies` metodo AmazonIdentityManagementClient's. È necessario un ListAttachedRolePoliciesRequestoggetto che contenga il nome del ruolo per cui elencare le politiche.

Richiama `getAttachedPolicies` l'ListAttachedRolePoliciesResultoggetto restituito per ottenere l'elenco delle politiche allegate. I risultati possono essere troncati; se il `getIsTruncated` metodo dell'`ListAttachedRolePoliciesResult`oggetto restituisce`true`, chiama il `setMarker` metodo dell'`ListAttachedRolePoliciesRequest`oggetto e usalo per richiamare `listAttachedRolePolicies` nuovamente per ottenere il successivo batch di risultati.

Importazioni

```
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;
```

Codice

```
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());
}
```

Vedi l'esempio completo su. GitHub

Distaccare una policy del ruolo

Per scollegare una policy da un ruolo, chiamate il `detachRolePolicy` metodo AmazonIdentityManagementClient th's, fornendogli il nome del ruolo e l'ARN della policy in a. DetachRolePolicyRequest

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su. GitHub

Ulteriori informazioni

- Panoramica delle politiche IAM nella Guida IAM per l'utente.
- AWS Riferimento alle politiche IAM nella Guida IAM per l'utente.
- CreatePolicynel riferimento all'API IAM
- GetPolicynel riferimento all'API IAM
- AttachRolePolicynel riferimento all'API IAM
- ListAttachedRolePoliciesnel riferimento all'API IAM
- DetachRolePolicynel riferimento all'API IAM

Utilizzo dei certificati del server IAM

Per abilitare le connessioni HTTPS al tuo sito Web o alla tua applicazione AWS, devi disporre di un certificato del server SSL/TLS. È possibile utilizzare un certificato server fornito da AWS Certificate Manager o uno ottenuto da un provider esterno.

Ti consigliamo di utilizzare ACM per fornire, gestire e distribuire i certificati del server. Con ACM puoi richiedere un certificato, distribuirlo nelle tue AWS risorse e lasciare che ACM gestisca i rinnovi dei certificati per te. I certificati forniti da ACM sono gratuiti. Per ulteriori informazioni su ACM, consulta la ACM User Guide.

Ottenere un certificato del server

È possibile recuperare un certificato del server chiamando il `getServerCertificate` metodo AmazonIdentityManagementClient's e passandogli un certificato GetServerCertificateRequestcon il nome del certificato.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su. GitHub

Elencazione di certificati del server

Per elencare i certificati del tuo server, chiama il `listServerCertificates` metodo AmazonIdentityManagementClient's con un ListServerCertificatesRequest. Restituisce ListServerCertificatesResult.

Chiama il `getServerCertificateMetadataList` metodo dell'`ListServerCertificateResult`oggetto restituito per ottenere un elenco di ServerCertificateMetadataoggetti che puoi utilizzare per ottenere informazioni su ciascun certificato.

I risultati possono essere troncati; se il `getIsTruncated` metodo dell'`ListServerCertificateResult`oggetto restituisce`true`, chiama il `setMarker` metodo dell'`ListServerCertificatesRequest`oggetto e usalo per richiamare `listServerCertificates` nuovamente e ottenere il successivo batch di risultati.

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

Vedi l'esempio completo su. GitHub

Aggiornamento di un certificato del server

È possibile aggiornare il nome o il percorso di un certificato del server chiamando il `updateServerCertificate` metodo AmazonIdentityManagementClient's. Richiede un UpdateServerCertificateRequestoggetto impostato con il nome corrente del certificato del server e un nuovo nome o un nuovo percorso da utilizzare.

Importazioni

```
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;
```

Codice

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

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

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

Vedi l'esempio completo su GitHub.

Eliminazione di un certificato del server

Per eliminare un certificato del server, chiama il `deleteServerCertificate` metodo AmazonIdentityManagementClient's con un DeleteServerCertificateRequestcontenente il nome del certificato.

Importazioni

```
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;
```

Codice

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

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

DeleteServerCertificateResult response =
```

```
    iam.deleteServerCertificate(request);
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Utilizzo dei certificati server nella Guida per l' IAM utente
- GetServerCertificatenella guida di riferimento all'API IAM
- ListServerCertificatesnel riferimento all'API IAM
- UpdateServerCertificatenel riferimento all'API IAM
- DeleteServerCertificatenel riferimento all'API IAM
- Guida per l'utente di ACM

Lambda Esempi di utilizzo di AWS SDK per Java

Questa sezione fornisce esempi di programmazione Lambda utilizzando AWS SDK per Java.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ogni tecnica. Il codice di esempio completo è disponibile su GitHub. Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

Argomenti

- Funzioni di richiamo, elenco ed eliminazione Lambda

Funzioni di richiamo, elenco ed eliminazione Lambda

Questa sezione fornisce esempi di programmazione con il client Lambda di servizio utilizzando. AWS SDK per Java Per informazioni su come creare una Lambda funzione, vedere Come creare AWS Lambda funzioni.

Argomenti

- Richiamo di una funzione
- Elencare le funzioni

- Eliminare una funzione

Richiamo di una funzione

È possibile richiamare una Lambda funzione creando un AWSLambdaoggetto e richiamando il relativo `invoke` metodo. Create un InvokeRequestoggetto per specificare informazioni aggiuntive come il nome della funzione e il payload da passare alla funzione. Lambda I nomi delle funzioni vengono visualizzati come arn:aws:lambda:us-east- 1:555556330391:function:. HelloFunction È possibile recuperare il valore osservando la funzione in. Console di gestione AWS

Per passare i dati del payload a una funzione, richiamate il `withPayload` metodo dell'InvokeRequestoggetto e specificate una stringa in formato JSON, come illustrato nel seguente esempio di codice.

Importazioni

```
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;
```

Codice

Il seguente esempio di codice mostra come richiamare una funzione. Lambda

```
        String functionName = args[0];

        InvokeRequest invokeRequest = new InvokeRequest()
                .withFunctionName(functionName)
                .withPayload("{\n" +
                        " \"Hello \": \"Paris\",\n" +
                        " \"countryCode\": \"FR\"\n" +
                        "}");
        InvokeResult invokeResult = null;

        try {
```

```
            AWSLambda awsLambda = AWSLambdaClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(Regions.US_WEST_2).build();

            invokeResult = awsLambda.invoke(invokeRequest);

            String ans = new String(invokeResult.getPayload().array(),
StandardCharsets.UTF_8);

            //write out the return value
            System.out.println(ans);

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

        System.out.println(invokeResult.getStatusCode());
```

Vedi l'esempio completo su GitHub.

Elencare le funzioni

Costruisci un AWSLambdaoggetto e richiama il suo metodo. `listFunctions` Questo metodo restituisce un ListFunctionsResultoggetto. È possibile richiamare il `getFunctions` metodo di questo oggetto per restituire un elenco di FunctionConfigurationoggetti. È possibile scorrere l'elenco per recuperare informazioni sulle funzioni. Il seguente esempio di codice Java mostra come ottenere i nomi di ogni funzione.

Importazioni

```
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;
```

Codice

Il seguente esempio di codice Java mostra come recuperare un elenco di nomi di Lambda funzioni.

```
        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);
        }
```

Vedi l'esempio completo su GitHub.

Eliminare una funzione

Costruisci un AWSLambdaoggetto e invoca il suo metodo. `deleteFunction` Crea un DeleteFunctionRequestoggetto e passalo al `deleteFunction` metodo. Questo oggetto contiene informazioni quali il nome della funzione da eliminare. I nomi delle funzioni vengono visualizzati come arn:aws:lambda:us-east- 1:555556330391:function:. HelloFunction È possibile recuperare il valore osservando la funzione in. Console di gestione AWS

Importazioni

```
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;
```

Codice

Il seguente codice Java mostra come eliminare una Lambda funzione.

```
        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);
        }
```

Vedi l'esempio completo su GitHub.

Amazon Pinpoint Esempi di utilizzo di AWS SDK per Java

In questa sezione vengono forniti esempi di programmazione di Amazon Pinpoint utilizzando AWS SDK per Java.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ogni tecnica. Il codice di esempio completo è disponibile su GitHub. Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

Argomenti

- Creazione ed eliminazione di app in Amazon Pinpoint
- Creazione di endpoint in Amazon Pinpoint
- Creazione di segmenti in Amazon Pinpoint
- Creazione di campagne in Amazon Pinpoint
- Aggiornamento dei canali in Amazon Pinpoint

Creazione ed eliminazione di app in Amazon Pinpoint

Un'app è un Amazon Pinpoint progetto in cui definisci il pubblico di un'applicazione distinta e coinvolgi questo pubblico con messaggi personalizzati. Gli esempi in questa pagina mostrano come creare una nuova app o eliminarne una esistente.

Creare un'app

Crea una nuova app Amazon Pinpoint fornendo un nome di app all'CreateAppRequestoggetto e quindi passando quell'oggetto al `createApp` metodo AmazonPinpointClient's.

Importazioni

```
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;
```

Codice

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

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

Vedi l'esempio completo su GitHub.

Eliminare un'app

Per eliminare un'app, AmazonPinpointClient chiama la `deleteApp` richiesta con un DeleteAppRequestoggetto impostato con il nome dell'app da eliminare.

Importazioni

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

Codice

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

pinpoint.deleteApp(deleteRequest);
```

Guarda l'esempio completo su GitHub.

Ulteriori informazioni

- App nell' Amazon Pinpoint API Reference
- App nell' Amazon Pinpoint API Reference

Creazione di endpoint in Amazon Pinpoint

Un endpoint identifica in modo univoco un dispositivo utente al quale è possibile inviare notifiche push. Amazon Pinpoint Se l'app è abilitata al Amazon Pinpoint supporto, l'app registra automaticamente un endpoint ogni Amazon Pinpoint volta che un nuovo utente apre l'app. L'esempio seguente mostra come aggiungere un nuovo endpoint a livello di codice.

Creazione di un endpoint

Crea un nuovo endpoint in fornendo i dati dell'endpoint in Amazon Pinpoint un oggetto. EndpointRequest

Importazioni

```
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;
```

Codice

```
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);
```

Quindi crea un UpdateEndpointRequestoggetto con quell' EndpointRequest oggetto. Infine, passa l' UpdateEndpointRequest oggetto al `updateEndpoint` metodo AmazonPinpointClient's.

Codice

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

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

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Aggiungere Endpoint nella Guida per gli Amazon Pinpoint sviluppatori
- Endpoint nell'API Reference Amazon Pinpoint

Creazione di segmenti in Amazon Pinpoint

Un segmento di utenti rappresenta un sottoinsieme dei tuoi utenti basato su caratteristiche condivise, ad esempio l'ultima volta che un utente ha aperto la tua app o il dispositivo che utilizza. L'esempio seguente mostra come definire un segmento di utenti.

Creazione di un segmento

Crea un nuovo segmento Amazon Pinpoint definendo le dimensioni del segmento in un SegmentDimensionsoggetto.

Importazioni

```
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;
```

Codice

```
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);
```

Quindi imposta l'SegmentDimensionsoggetto in a WriteSegmentRequest, che a sua volta viene utilizzato per creare un CreateSegmentRequestoggetto. Quindi passa l' CreateSegmentRequest oggetto al `createSegment` metodo AmazonPinpointClient's.

Codice

```
WriteSegmentRequest writeSegmentRequest = new WriteSegmentRequest()
```

```
        .withName("MySegment").withDimensions(dimensions);

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

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

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Amazon Pinpoint Segmenti nella Guida per l' Amazon Pinpoint utente
- Creazione di segmenti nella Guida per gli sviluppatori Amazon Pinpoint
- Segmenti nell'API Reference Amazon Pinpoint
- Segmento nell' Amazon Pinpoint API Reference

Creazione di campagne in Amazon Pinpoint

Puoi utilizzare le campagne per aumentare il coinvolgimento tra la tua app e i tuoi utenti. Puoi creare una campagna per raggiungere un determinato segmento di utenti con messaggi personalizzati o promozioni speciali. Questo esempio dimostra come creare una nuova campagna standard che invii una notifica push personalizzata a un segmento specifico.

Creazione di una campagna

Prima di creare una nuova campagna, devi definire una pianificazione e un messaggio e impostare questi valori in un WriteCampaignRequestoggetto.

Importazioni

```
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;
```

Codice

```
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);
```

Quindi crea una nuova campagna Amazon Pinpoint WriteCampaignRequestfornendo la configurazione della campagna a un CreateCampaignRequestoggetto. Infine, passa l' CreateCampaignRequest oggetto al `createCampaign` metodo AmazonPinpointClient's.

Codice

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

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

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Amazon Pinpoint Campagne nella Guida Amazon Pinpoint per l'utente
- Creazione di campagne nella Guida per Amazon Pinpoint gli sviluppatori
- Campagne nell' Amazon Pinpoint API Reference

- Campagna nell' Amazon Pinpoint API Reference
- Attività della campagna nell' Amazon Pinpoint API Reference
- Versioni della campagna nell' Amazon Pinpoint API Reference
- Versione della campagna nell' Amazon Pinpoint API Reference

Aggiornamento dei canali in Amazon Pinpoint

Un canale definisce i tipi di piattaforme a cui è possibile recapitare messaggi. Questo esempio mostra come utilizzare il APNs canale per inviare un messaggio.

Aggiornare un canale

Abilita l'accesso a un canale Amazon Pinpoint fornendo un ID app e un oggetto di richiesta del tipo di canale che desideri aggiornare. Questo esempio aggiorna il APNs canale, che richiede l'oggetto APNSChannelRequest. Impostateli nel UpdateApnsChannelRequeste passate l'oggetto al `updateApnsChannel` metodo AmazonPinpointClient's.

Importazioni

```
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;
```

Codice

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

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

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Amazon Pinpoint Canali nella Guida Amazon Pinpoint per l'utente
- Canale ADM nell' Amazon Pinpoint API Reference
- APNs Canale nell' Amazon Pinpoint API Reference
- APNs Sandbox Channel Amazon Pinpoint API Reference
- APNs Canale VoIP nell'API Reference Amazon Pinpoint
- APNs VoIP Sandbox Channel API Reference Amazon Pinpoint
- Baidu Channel API Reference Amazon Pinpoint
- Canale di posta elettronica nell'API Reference Amazon Pinpoint
- Canale GCM nell' Amazon Pinpoint API Reference
- Canale SMS nell' Amazon Pinpoint API Reference

Amazon S3 Esempi di utilizzo di AWS SDK per Java

In questa sezione vengono forniti esempi di programmazione di Amazon S3 utilizzando AWS SDK per Java.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ogni tecnica. Il codice di esempio completo è disponibile su GitHub. Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

Argomenti

- Creazione, elenco ed eliminazione Amazon S3 di bucket
- Esecuzione di operazioni sugli Amazon S3 oggetti
- Gestione delle autorizzazioni di Amazon S3 accesso per bucket e oggetti
- Gestione dell'accesso ai Amazon S3 bucket utilizzando le policy dei bucket
- Utilizzo TransferManager per Amazon S3 le operazioni
- Configurazione di un Amazon S3 bucket come sito Web
- Usa la Amazon S3 crittografia lato client

Creazione, elenco ed eliminazione Amazon S3 di bucket

Ogni oggetto (file) contenuto Amazon S3 deve risiedere all'interno di un bucket, che rappresenta una raccolta (contenitore) di oggetti. Ogni bucket è conosciuto da una chiave (nome), che deve essere univoca. Per informazioni dettagliate sui bucket e sulla loro configurazione, consulta Lavorare con Amazon S3 i bucket nella Guida per l' Amazon Simple Storage Service utente.

> **Note**
>
> Best practice
> Ti consigliamo di abilitare la regola del AbortIncompleteMultipartUploadciclo di vita sui tuoi bucket. Amazon S3
> Questa regola impone di interrompere Amazon S3 i caricamenti in più parti che non vengono completati entro un determinato numero di giorni dall'avvio. Quando viene superato il limite di tempo impostato, Amazon S3 interrompe il caricamento e quindi elimina i dati di caricamento incompleti.
> Per ulteriori informazioni, consulta Lifecycle Configuration for a Bucket with Versioning nella Guida per l'utente. Amazon S3

> **Note**
>
> Questi esempi di codice presuppongono che tu abbia compreso il materiale contenuto in Using the AWS SDK per Java e che tu abbia configurato AWS le credenziali predefinite utilizzando le informazioni contenute in Configurazione delle AWS credenziali e Area per lo sviluppo.

Creare un bucket

Utilizza il metodo del client AmazonS3. `createBucket` Il nuovo Bucket viene restituito. Il `createBucket` metodo genererà un'eccezione se il bucket esiste già.

> **Note**
>
> Per verificare se esiste già un bucket prima di tentare di crearne uno con lo stesso nome, chiama il metodo. `doesBucketExist` Ritornerà `true` se il bucket esiste e in altro modo. `false`

Importazioni

```
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;
```

Codice

```
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;
```

Vedi l'esempio completo su. GitHub

Creazione di un elenco di bucket

Utilizza il metodo del client AmazonS3. `listBucket` In caso di successo, viene restituito un elenco di Bucket.

Importazioni

```
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;
```

Codice

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

Vedi l'esempio completo su. GitHub

Eliminazione di un bucket

Prima di poter eliminare un Amazon S3 bucket, devi assicurarti che sia vuoto, altrimenti si verificherà un errore. Se si dispone di un bucket con versione, è necessario eliminare anche tutti gli oggetti con versione associati al bucket.

> **Note**
>
> L'esempio completo include ciascuno di questi passaggi in ordine, fornendo una soluzione completa per l'eliminazione di un bucket e del relativo contenuto. Amazon S3

Argomenti

- Rimuovi oggetti da un bucket senza versione prima di eliminarlo
- Rimuovi oggetti da un bucket con versione prima di eliminarlo
- Eliminazione di un bucket vuoto

Rimuovi oggetti da un bucket senza versione prima di eliminarlo

Utilizza il `listObjects` metodo del client AmazonS3 per recuperare l'elenco di oggetti ed eliminarli ciascuno. `deleteObject`

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

Guarda l'esempio completo su. GitHub

Rimuovi oggetti da un bucket con versione prima di eliminarlo

Se utilizzi un bucket con versione, devi anche rimuovere tutte le versioni memorizzate degli oggetti nel bucket prima che il bucket possa essere eliminato.

Utilizzando uno schema simile a quello utilizzato per rimuovere gli oggetti all'interno di un bucket, rimuovi gli oggetti con versione utilizzando il `listVersions` metodo del client AmazonS3 per elencare tutti gli oggetti con versione e quindi eliminarli ciascuno. `deleteVersion`

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

Guarda l'esempio completo su. GitHub

Eliminazione di un bucket vuoto

Dopo aver rimosso gli oggetti da un bucket (compresi gli oggetti con versione), puoi eliminare il bucket stesso utilizzando il metodo del client AmazonS3. `deleteBucket`

Importazioni

```
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;
```

Codice

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

Guarda l'esempio completo su. GitHub

Esecuzione di operazioni sugli Amazon S3 oggetti

Un Amazon S3 oggetto rappresenta un file o una raccolta di dati. Ogni oggetto deve risiedere all'interno di un bucket.

> **Note**
>
> Questi esempi di codice presuppongono che l'utente comprenda il materiale contenuto in Using the AWS SDK per Java e che abbia configurato AWS le credenziali predefinite utilizzando le informazioni contenute in Configurazione delle AWS credenziali e Area per lo sviluppo.

Argomenti

- Caricamento di un oggetto
- Elenco di oggetti
- Donwload di un oggetto
- Copiare, spostare o rinominare oggetti
- Eliminazione di un oggetto
- Eliminare più oggetti contemporaneamente

Caricamento di un oggetto

Utilizza il `putObject` metodo del client AmazonS3, fornendo un nome di bucket, un nome chiave e un file da caricare. Il bucket deve esistere o si verificherà un errore.

Importazioni

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

Codice

```
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);
}
```

Vedi l'esempio completo su. GitHub

Elenco di oggetti

Per ottenere un elenco di oggetti all'interno di un bucket, utilizza il `listObjects` metodo del client AmazonS3, fornendo il nome di un bucket.

Il `listObjects` metodo restituisce un ObjectListingoggetto che fornisce informazioni sugli oggetti nel bucket. Per elencare i nomi degli oggetti (chiavi), usa il `getObjectSummaries` metodo per ottenere un elenco di ObjectSummary oggetti S3, ognuno dei quali rappresenta un singolo oggetto nel bucket. Quindi chiama il suo `getKey` metodo per recuperare il nome dell'oggetto.

Importazioni

```
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;
```

Codice

```
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());
}
```

Vedi l'esempio completo su. GitHub

Donwload di un oggetto

Utilizza il `getObject` metodo del client AmazonS3, passandogli il nome di un bucket e dell'oggetto da scaricare. In caso di successo, il metodo restituisce un oggetto S3Object. Il bucket e la chiave dell'oggetto specificati devono esistere, altrimenti si verificherà un errore.

È possibile ottenere il contenuto dell'oggetto `getObjectContent` chiamando`S3Object`. Ciò restituisce un S3 ObjectInputStream che si comporta come un oggetto Java `InputStream` standard.

L'esempio seguente scarica un oggetto da S3 e ne salva il contenuto in un file (utilizzando lo stesso nome della chiave dell'oggetto).

Importazioni

```
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;
```

Codice

```
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);
}
```

Vedi l'esempio completo su. GitHub

Copiare, spostare o rinominare oggetti

Puoi copiare un oggetto da un bucket all'altro utilizzando il metodo del client AmazonS3. `copyObject` Richiede il nome del bucket da cui copiare, l'oggetto da copiare e il nome del bucket di destinazione.

Importazioni

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Codice

```
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!");
```

Vedi l'esempio completo su. GitHub

> **Note**
>
> È possibile utilizzare DeleteObject per spostare o rinominare un oggetto, copiando prima l'oggetto `copyObject` con un nuovo nome (è possibile utilizzare lo stesso bucket sia come origine che come destinazione) e quindi eliminando l'oggetto dalla sua posizione precedente.

Eliminazione di un oggetto

Utilizzate il `deleteObject` metodo del client AmazonS3, passandogli il nome di un bucket e dell'oggetto da eliminare. Il bucket e la chiave dell'oggetto specificati devono esistere, altrimenti si verificherà un errore.

Importazioni

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Codice

```
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);
}
```

Vedi l'esempio completo su. GitHub

Eliminare più oggetti contemporaneamente

Utilizzando il `deleteObjects` metodo del client AmazonS3, puoi eliminare più oggetti dallo stesso bucket passandone i nomi al metodo link:sdk-for-java/.html. v1/reference/com/amazonaws/services/s3/model/DeleteObjectsRequest

Importazioni

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
```

Codice

```
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);
}
```

Guarda l'esempio completo su. GitHub

Gestione delle autorizzazioni di Amazon S3 accesso per bucket e oggetti

È possibile utilizzare gli elenchi di controllo degli accessi (ACLs) per Amazon S3 i bucket e gli oggetti per un controllo dettagliato delle risorse. Amazon S3

> **Note**
>
> Questi esempi di codice presuppongono che l'utente comprenda il materiale contenuto in Using the AWS SDK per Java e che abbia configurato AWS le credenziali predefinite utilizzando le informazioni contenute in Configurazione delle AWS credenziali e Area per lo sviluppo.

Ottieni l'elenco di controllo degli accessi per un bucket

Per ottenere l'ACL corrente per un bucket, chiama il *`getBucketAcl`* metodo di AmazonS3, passandogli il nome del bucket da interrogare. Questo metodo restituisce un oggetto. AccessControlList Per ottenere ogni concessione di accesso nell'elenco, chiama il relativo `getGrantsAsList` metodo, che restituirà un elenco Java standard di oggetti Grant.

Importazioni

```
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;
```

Codice

```
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);
}
```

Vedi l'esempio completo su GitHub.

Imposta l'elenco di controllo degli accessi per un bucket

Per aggiungere o modificare le autorizzazioni a un ACL per un bucket, chiama il metodo di AmazonS3. `setBucketAcl` Richiede un AccessControlListoggetto che contenga un elenco di assegnatari e livelli di accesso da impostare.

Importazioni

```
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;
```

Codice

```
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**
>
> Puoi fornire l'identificatore univoco del beneficiario direttamente utilizzando la classe Grantee oppure utilizzare la EmailAddressGranteeclasse per impostare il beneficiario via e-mail, come abbiamo fatto qui.

Vedi GitHub l'esempio completo su.

Ottieni la lista di controllo degli accessi per un oggetto

Per ottenere l'ACL corrente per un oggetto, chiama il *`getObjectAcl`* metodo di AmazonS3, passandogli il nome del bucket e il nome dell'oggetto da interrogare. Ad esempio`getBucketAcl`, questo metodo restituisce un AccessControlListoggetto che puoi usare per esaminare ogni Grant.

Importazioni

```
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;
```

Codice

```
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);
}
```

Vedi l'esempio completo su GitHub.

Impostare l'elenco di controllo degli accessi per un oggetto

Per aggiungere o modificare le autorizzazioni a un ACL per un oggetto, chiamate il metodo di AmazonS3. `setObjectAcl` Richiede un AccessControlListoggetto che contenga un elenco di assegnatari e livelli di accesso da impostare.

Importazioni

```
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;
```

Codice

```
    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**
>
> Puoi fornire l'identificatore univoco del beneficiario direttamente utilizzando la classe Grantee oppure utilizzare la EmailAddressGranteeclasse per impostare il beneficiario via e-mail, come abbiamo fatto qui.

Vedi GitHub l'esempio completo su.

Ulteriori informazioni

- GET Bucket acl nell'API Reference Amazon S3
- INSERISCI Bucket acl nel riferimento API Amazon S3
- GET Object acl nel riferimento API Amazon S3
- INSERISCI Object acl nel riferimento API Amazon S3

Gestione dell'accesso ai Amazon S3 bucket utilizzando le policy dei bucket

Puoi impostare, ottenere o eliminare una policy sui bucket per gestire l'accesso ai tuoi Amazon S3 bucket.

Imposta una Bucket Policy

Puoi impostare la policy sui bucket per un determinato bucket S3 nei seguenti modi:

- Chiamando il client AmazonS3 e fornendogli un `setBucketPolicy` SetBucketPolicyRequest
- Impostazione diretta della policy utilizzando l'`setBucketPolicy`overload che richiede il nome del bucket e il testo della policy (in formato JSON)

Importazioni

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
```

Codice

```
    s3.setBucketPolicy(bucket_name, policy_text);
```

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

Utilizzate la classe di policy per generare o convalidare una policy

Quando fornisci una policy bucket a`setBucketPolicy`, puoi fare quanto segue:

- Specificate la policy direttamente come stringa di testo in formato JSON
- Crea la politica utilizzando la classe Policy

Utilizzando la `Policy` classe, non devi preoccuparti di formattare correttamente la stringa di testo. Per ottenere il testo della policy JSON dalla `Policy` classe, usa il suo `toJson` metodo.

Importazioni

```
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;
```

Codice

```
        new Statement(Statement.Effect.Allow)
                .withPrincipals(Principal.AllUsers)
                .withActions(S3Actions.GetObject)
                .withResources(new Resource(
                        "{region-arn}s3:::" + bucket_name + "/*")));
return bucket_policy.toJson();
```

La `Policy` classe fornisce anche un `fromJson` metodo che può tentare di creare una politica utilizzando una stringa JSON passata. Il metodo lo convalida per garantire che il testo possa essere trasformato in una struttura di policy valida e fallirà con un `IllegalArgumentException` se il testo della policy non è valido.

```
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());
}
```

È possibile utilizzare questa tecnica per preconvalidare una politica letta da un file o in altri modi.

Vedi l'esempio completo su. GitHub

Ottieni una Bucket Policy

Per recuperare la policy relativa a un Amazon S3 bucket, chiama il `getBucketPolicy` metodo del client AmazonS3, passandogli il nome del bucket da cui ottenere la policy.

Importazioni

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

Codice

```
  try {
      BucketPolicy bucket_policy = s3.getBucketPolicy(bucket_name);
      policy_text = bucket_policy.getPolicyText();
  } catch (AmazonServiceException e) {
      System.err.println(e.getErrorMessage());
      System.exit(1);
  }
```

Se il bucket denominato non esiste, se non hai accesso ad esso o se non ha una politica relativa al bucket, ne viene generata una. `AmazonServiceException`

Vedi l'esempio completo su. GitHub

Eliminare una policy del bucket

Per eliminare una policy sui bucket, chiama il client AmazonS3`deleteBucketPolicy`, fornendogli il nome del bucket.

Importazioni

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

Codice

```
try {
    s3.deleteBucketPolicy(bucket_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Questo metodo funziona anche se il bucket non dispone già di una policy. Se specifichi un nome di bucket che non esiste o se non hai accesso al bucket, viene generato un. `AmazonServiceException`

Vedi l'esempio completo su. GitHub

Ulteriori informazioni

- Accedi alla panoramica del linguaggio delle politiche nella Guida Amazon Simple Storage Service per l'utente
- Esempi di policy relative a Bucket nella Guida per l' Amazon Simple Storage Service utente

Utilizzo TransferManager per Amazon S3 le operazioni

È possibile utilizzare la AWS SDK per Java TransferManager classe per trasferire in modo affidabile file dall'ambiente locale Amazon S3 e copiare oggetti da una posizione S3 a un'altra. `TransferManager`può visualizzare lo stato di avanzamento di un trasferimento e mettere in pausa o riprendere i caricamenti e i download.

> **Note**
>
> Best practice
> Ti consigliamo di abilitare la regola del AbortIncompleteMultipartUploadciclo di vita nei tuoi bucket. Amazon S3
> Questa regola impone di interrompere Amazon S3 i caricamenti in più parti che non vengono completati entro un determinato numero di giorni dall'avvio. Quando viene superato il limite di

tempo impostato, Amazon S3 interrompe il caricamento e quindi elimina i dati di caricamento incompleti.
Per ulteriori informazioni, consulta Lifecycle Configuration for a Bucket with Versioning nella Guida per l'utente. Amazon S3

> **Note**
>
> Questi esempi di codice presuppongono che tu abbia compreso il materiale contenuto in Using the AWS SDK per Java e che tu abbia configurato AWS le credenziali predefinite utilizzando le informazioni contenute in Configurazione delle AWS credenziali e Area per lo sviluppo.

Carica file e directory

TransferManager puoi caricare file, elenchi di file e directory in qualsiasi Amazon S3 bucket che hai creato in precedenza.

Argomenti

- Carica un singolo file
- Carica un elenco di file
- Carica una directory

Carica un singolo file

TransferManagerIl `upload` metodo di Call, che fornisce un nome di Amazon S3 bucket, un nome di chiave (oggetto) e un oggetto Java File standard che rappresenta il file da caricare.

Importazioni

```
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;
```

Codice

```
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();
```

Il `upload` metodo ritorna immediatamente, fornendo un `Upload` oggetto da utilizzare per controllare lo stato del trasferimento o per attendere il completamento.

Vedi Attendi il completamento di un trasferimento per informazioni sull'utilizzo del `shutdownNow` metodo `waitForCompletion` per completare correttamente un trasferimento prima TransferManager di chiamare. In attesa del completamento del trasferimento, puoi interrogare o ascoltare gli aggiornamenti sullo stato e sui progressi del trasferimento. Per ulteriori informazioni, consulta Ottieni lo stato e l'avanzamento del trasferimento.

Guarda l'esempio completo su GitHub.

Carica un elenco di file

Per caricare più file in un'unica operazione, chiamate il TransferManager `uploadFileList` metodo, fornendo quanto segue:

- Un nome di Amazon S3 bucket
- Un prefisso chiave da aggiungere ai nomi degli oggetti creati (il percorso all'interno del bucket in cui posizionare gli oggetti)
- Un oggetto File che rappresenta la directory relativa da cui creare i percorsi dei file
- Un oggetto List contenente un set di oggetti File da caricare

Importazioni

```
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;
```

Codice

```
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();
```

Vedi Attendi il completamento di un trasferimento per informazioni sull'utilizzo del `shutdownNow` metodo `waitForCompletion` per completare correttamente un trasferimento prima TransferManager di chiamare. In attesa del completamento del trasferimento, puoi interrogare o ascoltare gli aggiornamenti sullo stato e sui progressi del trasferimento. Per ulteriori informazioni, consulta Ottieni lo stato e l'avanzamento del trasferimento.

L'MultipleFileUploadoggetto restituito da `uploadFileList` può essere utilizzato per interrogare lo stato o l'avanzamento del trasferimento. Per ulteriori informazioni, consulta Esamina lo stato di

avanzamento corrente di un trasferimento e Ottieni lo stato di avanzamento del trasferimento con a. ProgressListener

Puoi anche usare `MultipleFileUpload` il `getSubTransfers` metodo S per ottenere i singoli `Upload` oggetti per ogni file da trasferire. Per ulteriori informazioni, consulta Get the Progress of Subtransfer.

Vedi l'esempio completo su. GitHub

Carica una directory

È possibile utilizzare TransferManager il `uploadDirectory` metodo S per caricare un'intera directory di file, con l'opzione di copiare i file nelle sottodirectory in modo ricorsivo. Fornisci un nome di Amazon S3 bucket, un prefisso chiave S3, un oggetto File che rappresenta la directory locale da copiare e `boolean` un valore che indica se vuoi copiare le sottodirectory in modo ricorsivo (vero o falso).

Importazioni

```
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;
```

Codice

```
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();
```

Vedi Wait for a Transfer to Complete per informazioni su come utilizzare il metodo To `waitForCompletion` Completare con successo un trasferimento prima di chiamare. TransferManager `shutdownNow` In attesa del completamento del trasferimento, puoi interrogare o ascoltare gli aggiornamenti sullo stato e sui progressi del trasferimento. Per ulteriori informazioni, consulta Ottieni lo stato e l'avanzamento del trasferimento.

L'MultipleFileUploadoggetto restituito da `uploadFileList` può essere utilizzato per interrogare lo stato o l'avanzamento del trasferimento. Per ulteriori informazioni, consulta Esamina lo stato di avanzamento corrente di un trasferimento e Ottieni lo stato di avanzamento del trasferimento con a. ProgressListener

Puoi anche usare `MultipleFileUpload` il `getSubTransfers` metodo S per ottenere i singoli `Upload` oggetti per ogni file da trasferire. Per ulteriori informazioni, consulta Get the Progress of Subtransfer.

Vedi l'esempio completo su. GitHub

Scarica file o directory

Usa la TransferManager classe per scaricare un singolo file (Amazon S3 oggetto) o una directory (un nome di Amazon S3 bucket seguito dal prefisso di un oggetto) da. Amazon S3

Argomenti

- Scarica un singolo file
- Scarica una directory

Scarica un singolo file

Utilizzate il `download` metodo TransferManager's, fornendo il nome del Amazon S3 bucket contenente l'oggetto che desiderate scaricare, il nome della chiave (oggetto) e un oggetto File che rappresenta il file da creare sul sistema locale.

Importazioni

```
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;
```

Codice

```
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();
```

Vedi Wait for a Transfer to Complete per informazioni sull'utilizzo del `shutdownNow` metodo `waitForCompletion` per completare correttamente un trasferimento prima TransferManager di chiamare. In attesa del completamento del trasferimento, puoi interrogare o ascoltare gli aggiornamenti sullo stato e sui progressi del trasferimento. Per ulteriori informazioni, consulta Ottieni lo stato e l'avanzamento del trasferimento.

Guarda l'esempio completo su GitHub.

Scarica una directory

Per scaricare un set di file che condividono un prefisso di chiave comune (analogo a una directory su un file system) da Amazon S3, utilizzate il metodo. TransferManager `downloadDirectory` Il metodo utilizza il nome del Amazon S3 bucket contenente gli oggetti da scaricare, il prefisso dell'oggetto condiviso da tutti gli oggetti e un oggetto File che rappresenta la directory in cui scaricare i file sul sistema locale. Se la directory denominata non esiste ancora, verrà creata.

Importazioni

```
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;
```

Codice

```
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();
```

Vedi Attendi il completamento di un trasferimento per informazioni sull'utilizzo del `shutdownNow` metodo `waitForCompletion` per completare correttamente un trasferimento prima TransferManager di chiamare. In attesa del completamento del trasferimento, puoi interrogare o ascoltare gli aggiornamenti sullo stato e sui progressi del trasferimento. Per ulteriori informazioni, consulta Ottieni lo stato e l'avanzamento del trasferimento.

Guarda l'esempio completo su GitHub.

Copia oggetti

Per copiare un oggetto da un bucket S3 a un altro, usa il TransferManager `copy` metodo.

Importazioni

```
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;
```

Codice

```
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();
```

Vedi l'esempio completo su. GitHub

Attendi il completamento del trasferimento

Se l'applicazione (o il thread) riesce a bloccarsi fino al completamento del trasferimento, puoi utilizzare il `waitForCompletion` metodo dell'interfaccia Transfer per bloccare fino al completamento del trasferimento o al verificarsi di un'eccezione.

```
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);
}
```

È possibile visualizzare l'avanzamento dei trasferimenti se si effettuano sondaggi sugli eventi prima della chiamata`waitForCompletion`, si implementa un meccanismo di polling su un thread

separato o si ricevono aggiornamenti sullo stato di avanzamento in modo asincrono utilizzando un. ProgressListener

Vedi l'esempio completo su. GitHub

Ottieni lo stato e l'avanzamento del trasferimento

Ciascuna delle classi restituite dai `copy` metodi TransferManager `upload*download*`, e restituisce un'istanza di una delle seguenti classi, a seconda che si tratti di un'operazione a file singolo o multiplo.

Classe	Restituito da
Copia	`copy`
Scarica	`download`
MultipleFileDownload	`downloadDirectory`
Caricamento	`upload`
MultipleFileUpload	`uploadFileList` , `uploadDirectory`

Tutte queste classi implementano l'interfaccia Transfer. `Transfer`fornisce metodi utili per conoscere lo stato di avanzamento di un trasferimento, mettere in pausa o riprendere il trasferimento e ottenere lo stato attuale o finale del trasferimento.

Argomenti

- Verifica lo stato di avanzamento attuale di un trasferimento
- Ottieni Transfer Progress con un ProgressListener
- Ottieni lo stato di avanzamento dei subtrasferimenti

Verifica lo stato di avanzamento attuale di un trasferimento

Questo ciclo stampa lo stato di avanzamento di un trasferimento, ne esamina l'avanzamento corrente durante l'esecuzione e, una volta completato, ne stampa lo stato finale.

Importazioni

```
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;
```

Codice

```
// 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);
```

Vedi l'esempio completo su. GitHub

Ottieni Transfer Progress con un ProgressListener

Puoi allegare un ProgressListenera qualsiasi trasferimento utilizzando il `addProgressListener` metodo dell'interfaccia di trasferimento.

A ProgressListenerrichiede un solo metodo`progressChanged`, che accetta un ProgressEventoggetto. È possibile utilizzare l'oggetto per ottenere i byte totali dell'operazione chiamando il relativo `getBytes` metodo e il numero di byte trasferiti finora mediante la chiamata. `getBytesTransferred`

Importazioni

```
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;
```

Codice

```
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();
```

Vedi l'esempio completo su. GitHub

Ottieni lo stato di avanzamento dei subtrasferimenti

La MultipleFileUploadclasse può restituire informazioni sui suoi subtrasferimenti chiamando il suo `getSubTransfers` metodo. Restituisce una raccolta non modificabile di oggetti Upload che forniscono lo stato e l'avanzamento individuali del trasferimento di ogni sottotrasferimento.

Importazioni

```
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;
```

Codice

```
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);
```

Vedi l'esempio completo su. GitHub

Ulteriori informazioni

- Object Keys nella Guida Amazon Simple Storage Service per l'utente

Configurazione di un Amazon S3 bucket come sito Web

Puoi configurare un Amazon S3 bucket in modo che si comporti come un sito Web. Per fare ciò, è necessario impostare la configurazione del sito Web.

> **Note**
>
> Questi esempi di codice presuppongono che l'utente comprenda il materiale contenuto in Using the AWS SDK per Java e che abbia configurato AWS le credenziali predefinite utilizzando le informazioni contenute in Configurazione AWS delle credenziali e della regione per lo sviluppo.

Imposta la configurazione del sito Web di Bucket

Per impostare la configurazione del sito Web di un Amazon S3 bucket, chiama il `setWebsiteConfiguration` metodo di AmazonS3 con il nome del bucket per cui impostare la configurazione e un BucketWebsiteConfigurationoggetto contenente la configurazione del sito Web del bucket.

L'impostazione di un documento indice è obbligatoria; tutti gli altri parametri sono facoltativi.

Importazioni

```
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;
```

Codice

```
    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**
>
> L'impostazione della configurazione di un sito Web non modifica le autorizzazioni di accesso per il bucket. Per rendere visibili i file sul Web, è inoltre necessario impostare una policy relativa ai bucket che consenta l'accesso pubblico in lettura ai file contenuti nel bucket. Per ulteriori informazioni, consulta Gestione dell'accesso ai Amazon S3 bucket tramite le policy dei bucket.

Vedi l'esempio completo su. GitHub

Configurazione del sito Web di Get a Bucket

Per ottenere la configurazione del sito Web di un Amazon S3 bucket, chiama il `getWebsiteConfiguration` metodo di AmazonS3 con il nome del bucket per cui recuperare la configurazione.

La configurazione verrà restituita come oggetto. BucketWebsiteConfiguration Se non esiste una configurazione del sito Web per il bucket, `null` verrà restituita.

Importazioni

```
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;
```

Codice

```
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);
}
```

Vedi l'esempio completo su. GitHub

Elimina la configurazione del sito Web di Bucket

Per eliminare la configurazione del sito Web di un Amazon S3 bucket, chiama il `deleteWebsiteConfiguration` metodo di AmazonS3 con il nome del bucket da cui eliminare la configurazione.

Importazioni

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

Codice

```
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);
}
```

Guarda l'esempio completo su. GitHub

Ulteriori informazioni

- Sito web PUT Bucket nell' Amazon S3 API Reference
- Il sito web GET Bucket nell'API Reference Amazon S3
- Il sito web DELETE Bucket nell'API Reference Amazon S3

Usa la Amazon S3 crittografia lato client

La crittografia dei dati utilizzando il client di Amazon S3 crittografia è un modo per fornire un ulteriore livello di protezione per le informazioni sensibili in cui sono archiviate. Amazon S3 Gli esempi in questa sezione mostrano come creare e configurare il client di Amazon S3 crittografia per l'applicazione.

Se non conosci la crittografia, consulta le nozioni di base sulla crittografia nella Guida per sviluppatori AWS KMS per una panoramica di base dei termini e degli algoritmi di crittografia. Per informazioni sul supporto della crittografia in generale AWS SDKs, consulta AWS SDK Support for Amazon S3 Client-Side Encryption nella Guida generale. Amazon Web Services

> **Note**
>
> Questi esempi di codice presuppongono che tu abbia compreso il materiale contenuto in Using the AWS SDK per Java e che tu abbia configurato AWS le credenziali predefinite utilizzando le informazioni contenute in Configurazione delle AWS credenziali e Area per lo sviluppo.

Se si utilizza la versione 1.11.836 o precedente di AWS SDK per Java, vedere Amazon S3 Encryption Client Migration per informazioni sulla migrazione delle applicazioni alle versioni successive. Se non riesci a eseguire la migrazione, guarda questo esempio completo su. GitHub

Altrimenti, se utilizzi la versione 1.11.837 o successiva di AWS SDK per Java, esplora gli argomenti di esempio elencati di seguito per utilizzare la crittografia lato client. Amazon S3

Argomenti

- Amazon S3 crittografia lato client con chiavi master client
- Amazon S3 crittografia lato client con chiavi gestite AWS KMS

Amazon S3 crittografia lato client con chiavi master client

Gli esempi seguenti utilizzano la classe AmazonS3 EncryptionClient V2Builder per creare un client con crittografia lato client abilitata. Amazon S3 Una volta abilitata, tutti gli oggetti su cui carichi utilizzando questo client verranno crittografati. Amazon S3 Tutti gli oggetti ottenuti Amazon S3 utilizzando questo client verranno automaticamente decrittografati.

> **Note**
>
> Gli esempi seguenti dimostrano l'utilizzo della crittografia Amazon S3 lato client con chiavi master client gestite dal cliente. Per informazioni su come utilizzare la crittografia con chiavi gestite AWS KMS, consulta la sezione Crittografia lato Amazon S3 client con chiavi gestite KMS. AWS

Puoi scegliere tra due modalità di crittografia quando abiliti la Amazon S3 crittografia lato client: autenticata rigorosa o autenticata. Le sezioni seguenti mostrano come abilitare ogni tipo. Per informazioni sugli algoritmi utilizzati da ciascuna modalità, consultate la CryptoModedefinizione.

Importazioni obbligatorie

Importa le seguenti classi per questi esempi.

Importazioni

```
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;
```

Crittografia autenticata rigorosa

La crittografia autenticata rigorosa è la modalità predefinita se non `CryptoMode` viene specificata alcuna.

Per abilitare esplicitamente questa modalità, specificate il `StrictAuthenticatedEncryption` valore nel `withCryptoConfiguration` metodo.

> **Note**
>
> Per utilizzare la crittografia autenticata lato client, è necessario includere il file jar di Bouncy Castle più recente nel classpath dell'applicazione.

Codice

```
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");
```

Modalità di crittografia autenticata

Quando si utilizza `AuthenticatedEncryption` la modalità, durante la crittografia viene applicato un algoritmo di key wrapping migliorato. Durante la decrittografia in questa modalità, l'algoritmo può verificare l'integrità dell'oggetto decrittografato e generare un'eccezione se il controllo fallisce. Per ulteriori dettagli su come funziona la crittografia autenticata, consultate il post sul blog Amazon S3 Client-Side Authenticated Encryption.

> **Note**
>
> Per utilizzare la crittografia autenticata lato client, è necessario includere il file jar Bouncy Castle più recente nel classpath dell'applicazione.

Per abilitare questa modalità, specificate il valore nel metodo. `AuthenticatedEncryption` `withCryptoConfiguration`

Codice

```
AmazonS3EncryptionV2 s3EncryptionClientV2 =
 AmazonS3EncryptionClientV2Builder.standard()
        .withRegion(Regions.DEFAULT_REGION)
        .withClientConfiguration(new ClientConfiguration())
        .withCryptoConfiguration(new
 CryptoConfigurationV2().withCryptoMode(CryptoMode.AuthenticatedEncryption))
        .withEncryptionMaterialsProvider(new StaticEncryptionMaterialsProvider(new
 EncryptionMaterials(secretKey)))
        .build();

s3EncryptionClientV2.putObject(bucket_name, ENCRYPTED_KEY1, "This is the 1st content to
 encrypt");
```

Amazon S3 crittografia lato client con chiavi gestite AWS KMS

Gli esempi seguenti utilizzano la classe AmazonS3 EncryptionClient V2Builder per creare un client con crittografia lato client abilitata. Amazon S3 Una volta configurato, tutti gli oggetti su cui carichi

utilizzando questo client verranno crittografati. Amazon S3 Tutti gli oggetti ottenuti Amazon S3 utilizzando questo client vengono decrittografati automaticamente.

> **Note**
>
> Gli esempi seguenti mostrano come utilizzare la crittografia Amazon S3 lato client con AWS le chiavi gestite da KMS. Per informazioni su come utilizzare la crittografia con le proprie chiavi, vedere Crittografia Amazon S3 lato client con chiavi master client.

È possibile scegliere tra due modalità di crittografia quando si abilita la Amazon S3 crittografia lato client: autenticata rigorosa o autenticata. Le sezioni seguenti mostrano come abilitare ogni tipo. Per informazioni sugli algoritmi utilizzati da ciascuna modalità, consultate la CryptoModedefinizione.

Importazioni obbligatorie

Importa le seguenti classi per questi esempi.

Importazioni

```
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;
```

Crittografia autenticata rigorosa

La crittografia autenticata rigorosa è la modalità predefinita se non `CryptoMode` viene specificata alcuna.

Per abilitare esplicitamente questa modalità, specificate il `StrictAuthenticatedEncryption` valore nel `withCryptoConfiguration` metodo.

> **Note**
>
> Per utilizzare la crittografia autenticata lato client, è necessario includere il file jar di Bouncy Castle più recente nel classpath dell'applicazione.

Codice

```
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));
```

Richiamate il `putObject` metodo sul client di crittografia per caricare oggetti. Amazon S3

Codice

```
s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
 with a key created in the {console}");
```

È possibile recuperare l'oggetto utilizzando lo stesso client. Questo esempio chiama il `getObjectAsString` metodo per recuperare la stringa che è stata memorizzata.

Codice

```
System.out.println(s3Encryption.getObjectAsString(bucket_name, ENCRYPTED_KEY3));
```

modalità di crittografia autenticata

Quando si utilizza `AuthenticatedEncryption` la modalità, durante la crittografia viene applicato un algoritmo di key wrapping migliorato. Durante la decrittografia in questa modalità, l'algoritmo può verificare l'integrità dell'oggetto decrittografato e generare un'eccezione se il controllo fallisce. Per ulteriori dettagli su come funziona la crittografia autenticata, consultate il post sul blog Amazon S3 Client-Side Authenticated Encryption.

> **Note**
>
> Per utilizzare la crittografia autenticata lato client, è necessario includere il file jar Bouncy Castle più recente nel classpath dell'applicazione.

Per abilitare questa modalità, specificate il valore nel metodo. `AuthenticatedEncryption` `withCryptoConfiguration`

Codice

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

Configurazione del client AWS KMS

Il client di Amazon S3 crittografia crea un AWS KMS client per impostazione predefinita, a meno che non ne venga specificato uno esplicitamente.

Per impostare la regione per questo client creato automaticamente, imposta il AWS KMS . `awsKmsRegion`

Codice

```
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();
```

In alternativa, è possibile utilizzare il proprio AWS KMS client per inizializzare il client di crittografia.

Codice

```
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 Esempi di utilizzo di AWS SDK per Java

In questa sezione vengono forniti esempi di programmazione di Amazon SQS utilizzando AWS SDK per Java.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ogni tecnica. Il codice di esempio completo è disponibile su GitHub. Da qui puoi scaricare un singolo file sorgente o clonare l'archivio localmente per ottenere tutti gli esempi da creare ed eseguire.

Argomenti

Utilizzo delle code di Amazon SQS messaggi

Una coda di messaggi è il contenitore logico utilizzato per inviare messaggi in modo affidabile. Amazon SQS Sono disponibili due tipi di code: standard e first-in, first-out (FIFO). Per ulteriori informazioni sulle code e sulle differenze tra questi tipi, consulta la Guida per gli Amazon SQS sviluppatori.

Questo argomento descrive come creare, elencare, eliminare e ottenere l'URL di una Amazon SQS coda utilizzando. AWS SDK per Java

Creare una coda

Utilizza il `createQueue` metodo del client AmazonSQS, fornendo un CreateQueueRequestoggetto che descrive i parametri della coda.

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

Puoi utilizzare la forma semplificata di`createQueue`, che richiede solo un nome di coda, per creare una coda standard.

```
sqs.createQueue("MyQueue" + new Date().getTime());
```

Vedi l'esempio completo su. GitHub

Code di elenchi

Per elencare le Amazon SQS code relative al tuo account, chiama il metodo del client AmazonSQS. `listQueues`

Importazioni

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

Codice

```
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);
}
```

L'utilizzo dell'*`listQueues`*overload senza parametri restituisce tutte le code. È possibile filtrare i risultati restituiti passandogli un `ListQueuesRequest` oggetto.

Importazioni

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

Codice

```
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);
}
```

Vedi l'esempio completo su GitHub.

Ottenere l'URL di una coda

Chiama il metodo del client AmazonSQS. `getQueueUrl`

Importazioni

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

Codice

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
String queue_url = sqs.getQueueUrl(QUEUE_NAME).getQueueUrl();
```

Guarda l'esempio completo su. GitHub

Eliminare una coda

Fornisci l'URL della coda al metodo del client AmazonSQS. `deleteQueue`

Importazioni

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

Codice

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

Guarda l'esempio completo su. GitHub

Ulteriori informazioni

- Come funzionano Amazon SQS le code nella Guida per gli Amazon SQS sviluppatori
- CreateQueuenell' Amazon SQS API Reference
- GetQueueUrlnell' Amazon SQS API Reference
- ListQueuesnell' Amazon SQS API Reference
- DeleteQueuesnell' Amazon SQS API Reference

Invio, ricezione ed eliminazione di Amazon SQS messaggi

Questo argomento descrive come inviare, ricevere ed eliminare Amazon SQS messaggi. I messaggi vengono sempre distribuiti tramite una coda SQS.

Inviare un messaggio

Aggiungi un singolo messaggio a una Amazon SQS coda chiamando il metodo del client AmazonSQS. `sendMessage` Fornisci un SendMessageRequestoggetto che contenga l'URL della coda, il corpo del messaggio e il valore di ritardo opzionale (in secondi).

Importazioni

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

Codice

```
SendMessageRequest send_msg_request = new SendMessageRequest()
        .withQueueUrl(queueUrl)
        .withMessageBody("hello world")
        .withDelaySeconds(5);
sqs.sendMessage(send_msg_request);
```

Vedi l'esempio completo su. GitHub

Invia più messaggi contemporaneamente

Puoi inviare più di un messaggio in un'unica richiesta. Per inviare più messaggi, utilizza il `sendMessageBatch` metodo del client AmazonSQS, che richiede un URL SendMessageBatchRequestcontenente l'URL della coda e un elenco di messaggi (ciascuno uno SendMessageBatchRequestEntrya) da inviare. Puoi anche impostare un valore di ritardo opzionale per messaggio.

Importazioni

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
import com.amazonaws.services.sqs.model.SendMessageBatchRequestEntry;
```

Codice

```
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);
```

Vedi l'esempio completo su GitHub.

Ricevi messaggi

Recupera tutti i messaggi attualmente in coda chiamando il `receiveMessage` metodo del client AmazonSQS e passandogli l'URL della coda. I messaggi vengono restituiti come un elenco di oggetti Message.

Importazioni

```
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
```

Codice

```
List<Message> messages = sqs.receiveMessage(queueUrl).getMessages();
```

Elimina i messaggi dopo la ricezione

Dopo aver ricevuto un messaggio e averne elaborato il contenuto, eliminalo dalla coda inviando l'handle di ricezione del messaggio e l'URL di coda al metodo del client AmazonSQS. `deleteMessage`

Codice

```
for (Message m : messages) {
    sqs.deleteMessage(queueUrl, m.getReceiptHandle());
}
```

Guarda l'esempio completo su. GitHub

Ulteriori informazioni

- Come funzionano Amazon SQS le code nella Guida per gli Amazon SQS sviluppatori
- SendMessagenell' Amazon SQS API Reference
- SendMessageBatchnell' Amazon SQS API Reference
- ReceiveMessagenell' Amazon SQS API Reference
- DeleteMessagenell' Amazon SQS API Reference

Abilitazione del polling lungo per le code di Amazon SQS messaggi

Amazon SQS utilizza il polling breve per impostazione predefinita, interrogando solo un sottoinsieme di server, sulla base di una distribuzione casuale ponderata, per determinare se sono disponibili messaggi da includere nella risposta.

Il polling prolungato aiuta a ridurre i costi di utilizzo Amazon SQS riducendo il numero di risposte vuote quando non ci sono messaggi disponibili da restituire in risposta a una richiesta inviata a una ReceiveMessage coda ed eliminando le false risposte vuote. Amazon SQS

> **Note**
>
> È possibile impostare una frequenza di polling lunga da 1 a 20 secondi.

Attivazione di sondaggi lunghi durante la creazione di una coda

Per abilitare il polling lungo durante la creazione di una Amazon SQS coda, imposta l'`ReceiveMessageWaitTimeSeconds`attributo sull'CreateQueueRequestoggetto prima di chiamare il metodo della classe AmazonSQS. `createQueue`

Importazioni

```
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;
```

Codice

```
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;
    }
}
```

Guarda l'esempio completo su. GitHub

Abilitazione del long polling su una coda esistente

Oltre ad abilitare il polling lungo durante la creazione di una coda, puoi attivarlo anche su una coda esistente `ReceiveMessageWaitTimeSeconds` impostando il metodo «SetQueueAttributesRequestbefore call» della classe AmazonSQS. `setQueueAttributes`

Importazioni

```
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Codice

```
SetQueueAttributesRequest set_attrs_request = new SetQueueAttributesRequest()
        .withQueueUrl(queue_url)
        .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");
sqs.setQueueAttributes(set_attrs_request);
```

GitHubGuarda l'esempio completo su.

Abilitazione del long polling alla ricezione del messaggio

Puoi abilitare il polling prolungato quando ricevi un messaggio impostando il tempo di attesa in secondi sul metodo ReceiveMessageRequestche fornisci alla classe AmazonSQS. `receiveMessage`

> **Note**
>
> Dovresti assicurarti che il timeout della richiesta del AWS cliente sia superiore al tempo massimo di sondaggio lungo (20 secondi) in modo che `receiveMessage` le tue richieste non scadano in attesa del prossimo evento di sondaggio!

Importazioni

```
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
```

Codice

```
ReceiveMessageRequest receive_request = new ReceiveMessageRequest()
        .withQueueUrl(queue_url)
        .withWaitTimeSeconds(20);
sqs.receiveMessage(receive_request);
```

Guarda l'esempio completo su. GitHub

Ulteriori informazioni

- Amazon SQS Sondaggi lunghi nella Guida per gli Amazon SQS sviluppatori
- CreateQueuenell' Amazon SQS API Reference
- ReceiveMessagenell' Amazon SQS API Reference
- SetQueueAttributesnell' Amazon SQS API Reference

Impostazione del timeout di visibilità in Amazon SQS

Quando un messaggio viene ricevuto in Amazon SQS, rimane in coda fino a quando non viene eliminato per garantire la ricezione. Un messaggio ricevuto, ma non eliminato, sarà disponibile nelle richieste successive dopo un determinato timeout di visibilità per evitare che il messaggio venga ricevuto più di una volta prima di poter essere elaborato ed eliminato.

> **Note**
>
> Quando si utilizzano code standard, il timeout di visibilità non è una garanzia contro la ricezione di un messaggio due volte. Se utilizzi una coda standard, assicurati che il codice sia in grado di gestire il caso in cui lo stesso messaggio sia stato recapitato più di una volta.

Impostazione del timeout di visibilità dei messaggi per un singolo messaggio

Quando hai ricevuto un messaggio, puoi modificarne il timeout di visibilità inserendo un codice di ricezione ChangeMessageVisibilityRequestche passi al metodo della classe AmazonSQS. `changeMessageVisibility`

Importazioni

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

Codice

```
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);
```

Guarda l'esempio completo su. GitHub

Impostazione del timeout di visibilità dei messaggi per più messaggi contemporaneamente

Per impostare il timeout di visibilità dei messaggi per più messaggi contemporaneamente, crea un elenco di ChangeMessageVisibilityBatchRequestEntryoggetti, ciascuno contenente una stringa ID univoca e un handle di ricezione. Quindi, passa l'elenco al metodo della classe Amazon SQS client. `changeMessageVisibilityBatch`

Importazioni

```
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;
```

Codice

```
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);
```

Guarda l'esempio completo su. GitHub

Ulteriori informazioni

- Visibility Timeout nella Guida per gli Amazon SQS sviluppatori
- SetQueueAttributesnell' Amazon SQS API Reference
- GetQueueAttributesnell' Amazon SQS API Reference
- ReceiveMessagenell' Amazon SQS API Reference

- ChangeMessageVisibilitynell' Amazon SQS API Reference
- ChangeMessageVisibilityBatchnell' Amazon SQS API Reference

Utilizzo di Dead Letter Queues in Amazon SQS

Amazon SQS fornisce supporto per le code di lettere morte. Una coda di lettere morte è una coda che altre code (di origine) possono indirizzare per i messaggi che non possono essere elaborati correttamente. Puoi riservare e isolare questi messaggi nella coda DLQ per determinare perché l'elaborazione non è riuscita.

Creazione di una coda di lettere morte

Una coda di lettere morte viene creata allo stesso modo di una coda normale, ma presenta le seguenti restrizioni:

- Una coda di lettere morte deve essere dello stesso tipo di coda (FIFO o standard) della coda di origine.
- È necessario creare una coda di lettere morte utilizzando la stessa regione Account AWS e della coda di origine.

Qui creiamo due Amazon SQS code identiche, una delle quali fungerà da coda di lettere morte:

Importazioni

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

Codice

```
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;
    }
}
```

Vedi l'esempio completo su. GitHub

Designazione di una coda di lettere morte per una coda di origine

Per designare una coda di lettere morte, è necessario innanzitutto creare una politica di redrive e quindi impostare la politica negli attributi della coda. Una politica di redrive è specificata in JSON e specifica l'ARN della coda di lettere morte e il numero massimo di volte in cui il messaggio può essere ricevuto e non elaborato prima di essere inviato alla coda di lettere morte.

Per impostare la politica di redrive per la coda di origine, chiama il `setQueueAttributes` metodo della classe AmazonSQS con un SetQueueAttributesRequestoggetto per il quale hai impostato l'attributo con la tua politica di redrive JSON. `RedrivePolicy`

Importazioni

```
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Codice

```
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);
```

Guarda l' GitHubesempio completo su.

Ulteriori informazioni

- Utilizzo di Amazon SQS Dead Letter Queues nella Guida per gli Amazon SQS sviluppatori
- SetQueueAttributesnell' Amazon SQS API Reference

Amazon SWF Esempi di utilizzo di AWS SDK per Java

Amazon SWFè un servizio di gestione del flusso di lavoro che aiuta gli sviluppatori a creare e scalare flussi di lavoro distribuiti che possono avere fasi parallele o sequenziali costituite da attività, flussi di lavoro secondari o persino attività Lambda.

Esistono due modi per utilizzare l' AWS SDK per Java, Amazon SWF utilizzando l'oggetto client SWF o utilizzando l'oggetto per Java. AWS Flow Framework Il AWS Flow Framework for Java è inizialmente più difficile da configurare, poiché fa un uso intensivo di annotazioni e si basa su librerie aggiuntive come AspectJ e Spring Framework. Tuttavia, per progetti di grandi dimensioni o complessi, risparmierai tempo di programmazione utilizzando for Java. AWS Flow Framework Per ulteriori informazioni, consulta la Guida AWS Flow Framework per gli sviluppatori di Java.

Questa sezione fornisce esempi di programmazione Amazon SWF utilizzando direttamente il AWS SDK per Java client.

Argomenti

- Nozioni di base su SWF
- Creazione di un' Amazon SWF applicazione semplice

- Lambda Compiti
- Chiusura graduale degli addetti alle attività e ai flussi di lavoro
- Registrazione di domini
- Elenco di domini

Nozioni di base su SWF

Questi sono schemi generali per l' Amazon SWF utilizzo di. AWS SDK per JavaÈ inteso principalmente come riferimento. Per un tutorial introduttivo più completo, vedi Creazione di un'applicazione semplice Amazon SWF.

Dipendenze

Amazon SWF Le applicazioni di base richiederanno le seguenti dipendenze, incluse in: AWS SDK per Java

- aws-java-sdk-1.12.*.jar
- commons-logging-1.2.*.jar
- http://client-4.3.*.jar
- http://core-4.3.*.jar
- jackson-annotazioni-2.12.*.jar
- jackson-core-2.12.*.jar
- jackson-databind-2.12.*.jar
- joda-time-2.8.*.jar

> **Note**
>
> I numeri di versione di questi pacchetti variano a seconda della versione dell'SDK in uso, ma le versioni fornite con l'SDK sono state testate per verificarne la compatibilità e sono quelle che dovresti usare.

AWS Flow Framework per le applicazioni Java richiedono una configurazione aggiuntiva e dipendenze aggiuntive. AWS Flow Framework Per ulteriori informazioni sull'utilizzo del framework, consulta la Java Developer Guide.

Importazioni

In generale, è possibile utilizzare le seguenti importazioni per lo sviluppo del codice:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Tuttavia, è buona norma importare solo le classi necessarie. Probabilmente finirai per specificare classi particolari nell'`com.amazonaws.services.simpleworkflow.model`area di lavoro:

```
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;
```

Se stai usando il AWS Flow Framework per Java, importerai le classi dall'area di lavoro. `com.amazonaws.services.simpleworkflow.flow` Per esempio:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.flow.ActivityWorker;
```

> **Note**
>
> The AWS Flow Framework for Java ha requisiti aggiuntivi oltre a quelli di base AWS SDK per Java. Per ulteriori informazioni, consulta la Guida AWS Flow Framework per gli sviluppatori di Java.

Utilizzo della classe client SWF

La vostra interfaccia di base Amazon SWF è tramite le AmazonSimpleWorkflowAsyncClientclassi AmazonSimpleWorkflowClientor. La differenza principale tra queste è che la `\*AsyncClient` classe restituisce oggetti Future per la programmazione concorrente (asincrona).

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

Creazione di un' Amazon SWF applicazione semplice

Questo argomento ti introdurrà alla programmazione di Amazon SWFapplicazioni con AWS SDK per Java, presentando alcuni concetti importanti lungo il percorso.

Informazioni sull'esempio

Il progetto di esempio creerà un flusso di lavoro con una singola attività che accetta i dati del flusso di lavoro trasferiti attraverso il AWS cloud (nella tradizione HelloWorld, sarà il nome di qualcuno da salutare) e quindi stamperà un messaggio di saluto in risposta.

Anche se a prima vista sembra molto semplice, Amazon SWF le applicazioni sono costituite da una serie di parti che lavorano insieme:

- Un dominio, utilizzato come contenitore logico per i dati di esecuzione del flusso di lavoro.
- Uno o più flussi di lavoro che rappresentano componenti di codice che definiscono l'ordine logico di esecuzione delle attività del flusso di lavoro e dei flussi di lavoro secondari.
- Un addetto al flusso di lavoro, noto anche come decisore, che effettua sondaggi sulle attività decisionali e pianifica le attività o i flussi di lavoro per bambini in risposta.
- Una o più attività, ognuna delle quali rappresenta un'unità di lavoro nel flusso di lavoro.
- Un addetto alle attività che analizza le attività svolte ed esegue metodi di attività in risposta.
- Uno o più elenchi di attività, ossia code gestite da chi lavora Amazon SWF per inviare richieste ai lavoratori del flusso di lavoro e delle attività. Le attività presenti in un elenco di attività destinato agli addetti al flusso di lavoro sono denominate attività decisionali. Quelle destinate agli addetti alle attività sono chiamate attività attive.
- Un programma di avvio del flusso di lavoro che avvia l'esecuzione del flusso di lavoro.

Dietro le quinte, Amazon SWF orchestra il funzionamento di questi componenti, ne coordina il flusso dal AWS cloud, trasferisce i dati tra di loro, gestisce i timeout e le notifiche Heartbeat e registra la cronologia di esecuzione del flusso di lavoro.

Prerequisiti

Ambiente di sviluppo

L'ambiente di sviluppo utilizzato in questo tutorial è composto da:

- Tipo AWS SDK per Java.

- Apache Maven (3.3.1).
- JDK 1.7 o successivo. Questo tutorial è stato sviluppato e testato utilizzando JDK 1.8.0.
- Un buon editor di testo Java (a tua scelta).

> **Note**
>
> Se utilizzi un sistema di compilazione diverso da Maven, puoi comunque creare un progetto utilizzando i passaggi appropriati per il tuo ambiente e utilizzare i concetti forniti qui per procedere. Ulteriori informazioni sulla configurazione e l'utilizzo di AWS SDK per Java con vari sistemi di compilazione sono disponibili in Guida introduttiva.
> Allo stesso modo, ma con maggiore impegno, i passaggi mostrati qui possono essere implementati utilizzando uno qualsiasi dei programmi AWS SDKs con supporto per Amazon SWF.

Tutte le dipendenze esterne necessarie sono incluse in AWS SDK per Java, quindi non c'è altro da scaricare.

AWS Accesso

Per completare correttamente questo tutorial, è necessario avere accesso al portale di AWS accesso come descritto nella sezione di configurazione di base di questa guida.

Le istruzioni descrivono come accedere alle credenziali temporanee da copiare e incollare nel `credentials` file condiviso locale. Le credenziali temporanee che incolli devono essere associate a un ruolo IAM AWS IAM Identity Center che dispone delle autorizzazioni per accedere ad Amazon SWF. Dopo aver incollato le credenziali temporanee, il `credentials` file avrà un aspetto simile al seguente.

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

Queste credenziali temporanee sono associate al profilo. `default`

Create un progetto SWF

1. Inizia un nuovo progetto con Maven:

```
mvn archetype:generate -DartifactId=helloswf \
-DgroupId=aws.example.helloswf -DinteractiveMode=false
```

Questo creerà un nuovo progetto con una struttura di progetto Maven standard:

```
helloswf
### pom.xml
### src
    ### main
    #   ### java
    #       ### aws
    #           ### example
    #               ### helloswf
    #                   ### App.java
    ### test
        ### ...
```

Puoi ignorare o eliminare la `test` directory e tutto ciò che contiene, non la useremo per questo tutorial. Puoi anche eliminarla`App.java`, poiché la sostituiremo con nuove classi.

2. Modifica il `pom.xml` file del progetto e aggiungi il aws-java-sdk-simpleworkflowmodulo aggiungendone una dipendenza all'interno del `<dependencies>` blocco.

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

3. Assicurati che Maven crei il tuo progetto con il supporto per JDK 1.7+. Aggiungi quanto segue al tuo progetto (prima o dopo il blocco) in: `<dependencies>` `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>
```

Codifica il progetto

Il progetto di esempio sarà composto da quattro applicazioni separate, che esamineremo una per una:

- HelloTypes.java --contiene i dati sul dominio, l'attività e il tipo di flusso di lavoro del progetto, condivisi con gli altri componenti. Gestisce anche la registrazione di questi tipi con SWF.
- ActivityWorker.java --contiene l'activity worker, che analizza le attività da svolgere ed esegue le attività in risposta.
- WorkflowWorker.java --contiene l'operatore del flusso di lavoro (decisore), che effettua sondaggi sulle attività decisionali e pianifica nuove attività.
- WorkflowStarter.java --contiene lo starter del flusso di lavoro, che avvia una nuova esecuzione del flusso di lavoro, che farà sì che SWF inizi a generare attività decisionali e di workflow per i lavoratori.

Procedure comuni per tutti i file sorgente

Tutti i file creati per ospitare le classi Java avranno alcune cose in comune. Per motivi di tempo, questi passaggi saranno impliciti ogni volta che aggiungerai un nuovo file al progetto:

1. Create il file nella `src/main/java/aws/example/helloswf/` cartella del progetto.
2. Aggiungi una `package` dichiarazione all'inizio di ogni file per dichiararne lo spazio dei nomi. Il progetto di esempio utilizza:

```
package aws.example.helloswf;
```

3. Aggiungi `import` dichiarazioni per la AmazonSimpleWorkflowClientclasse e per più classi nel `com.amazonaws.services.simpleworkflow.model` namespace. Per semplificare le cose, useremo:

```
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Registra un dominio, un flusso di lavoro e tipi di attività

Inizieremo creando una nuova classe eseguibile,`HelloTypes.java`. Questo file conterrà dati condivisi che diverse parti del flusso di lavoro dovranno conoscere, come il nome e la versione dell'attività e i tipi di flusso di lavoro, il nome di dominio e il nome dell'elenco delle attività.

1. Apri l'editor di testo e crea il file`HelloTypes.java`, aggiungendo una dichiarazione del pacchetto e le importazioni secondo i passaggi comuni.
2. Dichiara la `HelloTypes` classe e forniscile i valori da utilizzare per le attività registrate e i tipi di flusso di lavoro:

   ```
       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";
   ```

   Questi valori verranno utilizzati in tutto il codice.
3. Dopo le dichiarazioni String, create un'istanza della AmazonSimpleWorkflowClientclasse. Questa è l'interfaccia di base per i Amazon SWF metodi forniti da. AWS SDK per Java

   ```
   private static final AmazonSimpleWorkflow swf =

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

   Lo snippet precedente presuppone che le credenziali temporanee siano associate al profilo. `default` Se utilizzi un profilo diverso, modifica il codice precedente come segue e sostituiscilo *`profile_name`* con il nome del nome del profilo effettivo.

   ```
   private static final AmazonSimpleWorkflow swf =
           AmazonSimpleWorkflowClientBuilder
                   .standard()
                   .withCredentials(new ProfileCredentialsProvider("profile_name"))
                   .withRegion(Regions.DEFAULT_REGION)
   ```

```
                .build();
```

4. Aggiungete una nuova funzione per registrare un dominio SWF. Un dominio è un contenitore logico per una serie di attività SWF e tipi di workflow correlati. I componenti SWF possono comunicare tra loro solo se esistono all'interno dello stesso dominio.

```
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!");
}
```

Quando registrate un dominio, gli fornite un nome (qualsiasi set di 1-256 caratteri esclusi `:` i caratteri di controllo o la stringa letterale '`arn') e un periodo di conservazione, ossia il numero di giorni in cui Amazon SWF verranno conservati i dati della cronologia di esecuzione del flusso di lavoro dopo il completamento dell'esecuzione di un flusso di lavoro. `/ | Il` periodo massimo di conservazione dell'esecuzione del flusso di lavoro è di 90 giorni. Per ulteriori informazioni, consulta RegisterDomainRequest.

Se esiste già un dominio con quel nome, DomainAlreadyExistsExceptionviene generato un. Poiché non ci interessa se il dominio è già stato creato, possiamo ignorare l'eccezione.

> **Note**
>
> Questo codice dimostra uno schema comune quando si lavora con i AWS SDK per Java metodi, i dati per il metodo sono forniti da una classe nel `simpleworkflow.model` namespace, che viene istanziata e compilata utilizzando i metodi concatenabili. `0with*`

5. Aggiungi una funzione per registrare un nuovo tipo di attività. Un'attività rappresenta un'unità di lavoro nel flusso di lavoro.

```
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!");
    }
```

Un tipo di attività è identificato da un nome e da una versione, che vengono utilizzati per identificare in modo univoco l'attività rispetto a qualsiasi altra attività nel dominio in cui è registrata. Le attività contengono anche una serie di parametri opzionali, come l'elenco di attività predefinito utilizzato per ricevere attività e dati da SWF e una serie di timeout diversi che è possibile utilizzare per imporre vincoli sulla durata delle diverse parti dell'attività. Per ulteriori informazioni, consulta RegisterActivityTypeRequest.

> **Note**
>
> Tutti i valori di timeout sono specificati in secondi. Vedi Tipi di Amazon SWF timeout per una descrizione completa di come i timeout influiscono sulle esecuzioni del flusso di lavoro.

Se il tipo di attività che stai cercando di registrare esiste già, viene generato un TypeAlreadyExistsException. Aggiungi una funzione per registrare un nuovo tipo di flusso di lavoro. Un flusso di lavoro, noto anche come decisore, rappresenta la logica di esecuzione del flusso di lavoro.

+

```
    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!");
    }
```

+

Analogamente ai tipi di attività, i tipi di flusso di lavoro sono identificati da un nome e da una versione e dispongono anche di timeout configurabili. Per ulteriori informazioni, consulta RegisterWorkflowTypeRequest.

+

Se il tipo di flusso di lavoro che stai cercando di registrare esiste già, TypeAlreadyExistsExceptionviene generato un. Infine, rendi eseguibile la classe fornendole un `main` metodo che registrerà a turno il dominio, il tipo di attività e il tipo di flusso di lavoro:

+

```
    registerDomain();
    registerWorkflowType();
    registerActivityType();
```

Ora puoi creare ed eseguire l'applicazione per eseguire lo script di registrazione o continuare a codificare i lavoratori delle attività e del flusso di lavoro. Una volta registrati il dominio, il flusso di lavoro e l'attività, non sarà necessario eseguirli di nuovo: questi tipi persistono finché non li renderai obsoleti personalmente.

Implementa l'Activity Worker

Un'attività è l'unità di lavoro di base in un flusso di lavoro. Un flusso di lavoro fornisce la logica, la pianificazione delle attività da eseguire (o altre azioni da intraprendere) in risposta alle attività decisionali. Un tipico flusso di lavoro è in genere costituito da una serie di attività che possono essere eseguite in modo sincrono, asincrono o una combinazione di entrambe.

L'activity worker è il bit di codice che analizza le attività generate da Amazon SWF in risposta alle decisioni relative al flusso di lavoro. Quando riceve un'attività, esegue l'attività corrispondente e restituisce una risposta di successo/fallimento al flusso di lavoro.

Implementeremo un semplice Activity Worker che gestisca una singola attività.

1. Apri l'editor di testo e crea il file`ActivityWorker.java`, aggiungendo una dichiarazione del pacchetto e importandolo secondo i passaggi comuni.

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

2. Aggiungete la `ActivityWorker` classe al file e assegnategli un membro di dati che contenga un client SWF che useremo per interagire con Amazon SWF:

```
    private static final AmazonSimpleWorkflow swf =

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

3. Aggiungi il metodo che useremo come attività:

```
private static String sayHello(String input) throws Throwable {
    return "Hello, " + input + "!";
}
```

   L'attività prende semplicemente una stringa, la combina in un saluto e restituisce il risultato. Sebbene ci siano poche possibilità che questa attività generi un'eccezione, è consigliabile progettare attività che possano generare un errore se qualcosa va storto.

4. Aggiungi un `main` metodo che useremo come metodo di polling delle attività. Inizieremo aggiungendo del codice per esaminare l'elenco delle attività relative alle attività:

```
        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();
```

L'attività riceve le attività Amazon SWF richiamando il `pollForActivityTask` metodo del client SWF, specificando il dominio e l'elenco delle attività da utilizzare nel pass-in. PollForActivityTaskRequest

Una volta ricevuta un'operazione, ne richiamiamo un identificatore univoco richiamando il metodo dell'operazione. `getTaskToken`

5. Quindi, scrivi del codice per elaborare le attività che arrivano. Aggiungi quanto segue al tuo `main` metodo, subito dopo il codice che esegue il sondaggio per l'attività e ne recupera il token di attività.

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

Se il task token non lo è`null`, allora possiamo iniziare a eseguire il metodo activity (`sayHello`), fornendogli i dati di input inviati con l'attività.

Se l'operazione ha avuto esito positivo (non è stato generato alcun errore), l'operatore risponde a SWF chiamando il `respondActivityTaskCompleted` metodo del client SWF con un RespondActivityTaskCompletedRequestoggetto contenente il token dell'attività e i dati dei risultati dell'attività.

D'altra parte, se l'operazione ha esito negativo, rispondiamo chiamando il `respondActivityTaskFailed` metodo con un RespondActivityTaskFailedRequestoggetto, passandogli il token dell'operazione e le informazioni sull'errore.

> **Note**
>
> Questa attività non si interromperà correttamente se interrotta. Sebbene non rientri nello scopo di questo tutorial, un'implementazione alternativa di questo activity worker viene fornita nell'argomento di accompagnamento, Shutting Down Activity and Workflow Workers Gracefully.

Implementa il workflow worker

La logica del flusso di lavoro risiede in un pezzo di codice noto come workflow worker. L'addetto al flusso di lavoro esegue sondaggi sulle attività decisionali inviate Amazon SWF nel dominio e nell'elenco di attività predefinito con cui è stato registrato il tipo di flusso di lavoro.

Quando l'addetto al flusso di lavoro riceve un'attività, prende una sorta di decisione (in genere se pianificare o meno una nuova attività) e intraprende un'azione appropriata (come la pianificazione dell'attività).

1. Apri l'editor di testo e crea il file`WorkflowWorker.java`, aggiungendo una dichiarazione del pacchetto e le importazioni secondo i passaggi comuni.
2. Aggiungi alcune importazioni aggiuntive al file:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
```

3. Dichiarate la `WorkflowWorker` classe e create un'istanza della AmazonSimpleWorkflowClientclasse utilizzata per accedere ai metodi SWF.

```
    private static final AmazonSimpleWorkflow swf =

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

4. Aggiungete il metodo. `main` Il metodo si ripete continuamente, analizzando le attività decisionali utilizzando il metodo del client SWF. `pollForDecisionTask` PollForDecisionTaskRequestFornisce i dettagli.

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

   Una volta ricevuta un'attività, ne chiamiamo il `getTaskToken` metodo, che restituisce una stringa che può essere utilizzata per identificare l'attività. Se il token restituito non lo è`null`, lo elaboriamo ulteriormente nel `executeDecisionTask` metodo, passandogli il token dell'attività e l'elenco degli HistoryEventoggetti inviati con l'operazione.

5. Aggiungiamo il `executeDecisionTask` metodo, prendendo il task token (a`String`) e l'`HistoryEvent`elenco.

```
    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;
```

Abbiamo anche configurato alcuni membri dei dati per tenere traccia di cose come:

- Un elenco di oggetti Decision utilizzati per riportare i risultati dell'elaborazione dell'attività.
- Una stringa per contenere l'input del flusso di lavoro fornito dall'evento WorkflowExecutionStarted "»
- un conteggio delle attività pianificate e aperte (in esecuzione) per evitare di pianificare la stessa attività quando è già stata pianificata o è attualmente in esecuzione.
- un valore booleano per indicare che l'attività è stata completata.
- Una stringa per contenere i risultati dell'attività, per restituirli come risultato del nostro flusso di lavoro.

6. Quindi, aggiungi del codice `executeDecisionTask` per elaborare gli `HistoryEvent` oggetti inviati con l'attività, in base al tipo di evento riportato dal `getEventType` metodo.

```
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("]");
```

Ai fini del nostro flusso di lavoro, siamo particolarmente interessati a:

- l'evento WorkflowExecutionStarted "", che indica che l'esecuzione del flusso di lavoro è iniziata (in genere significa che è necessario eseguire la prima attività del flusso di lavoro) e che fornisce l'input iniziale fornito al flusso di lavoro. In questo caso, si tratta della parte relativa al nome del messaggio di saluto, quindi viene salvata in una stringa da utilizzare per pianificare l'esecuzione dell'attività.
- l'evento "ActivityTaskCompleted", che viene inviato una volta completata l'attività pianificata. I dati dell'evento includono anche il valore restituito dall'attività completata. Poiché abbiamo una sola attività, utilizzeremo quel valore come risultato dell'intero flusso di lavoro.

Gli altri tipi di eventi possono essere utilizzati se il flusso di lavoro li richiede. Vedi la descrizione della HistoryEventclasse per informazioni su ogni tipo di evento.

+ NOTA: le stringhe nelle `switch` istruzioni sono state introdotte in Java 7. Se stai utilizzando una versione precedente di Java, puoi utilizzare la EventTypeclasse per convertire il valore `String` restituito da `history_event.getType()` in un valore enum e poi tornare a, `String` se necessario:

```
EventType et = EventType.fromValue(event.getEventType());
```

1. Dopo l'`switch`istruzione, aggiungete altro codice per rispondere con una decisione appropriata in base all'attività ricevuta.

```
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);
```

- Se l'attività non è stata ancora programmata, rispondiamo con una `ScheduleActivityTask` decisione, che fornisce informazioni in una ScheduleActivityTaskDecisionAttributesstruttura sull'attività da programmare successivamente, inclusi anche i dati da Amazon SWF inviare all'attività. Amazon SWF
- Se l'attività è stata completata, consideriamo completato l'intero flusso di lavoro e rispondiamo con una `CompletedWorkflowExecution` decisione, compilando una CompleteWorkflowExecutionDecisionAttributesstruttura per fornire dettagli sul flusso di lavoro completato. In questo caso, restituiamo il risultato dell'attività.

In entrambi i casi, le informazioni sulla decisione vengono aggiunte all'`Decision`elenco dichiarato all'inizio del metodo.

2. Completa l'attività decisionale restituendo l'elenco degli `Decision` oggetti raccolti durante l'elaborazione dell'attività. Aggiungi questo codice alla fine del `executeDecisionTask` metodo che abbiamo scritto:

```
swf.respondDecisionTaskCompleted(
    new RespondDecisionTaskCompletedRequest()
        .withTaskToken(taskToken)
        .withDecisions(decisions));
```

Il `respondDecisionTaskCompleted` metodo del client SWF utilizza il token task che identifica l'operazione e l'elenco degli `Decision` oggetti.

Implementate lo starter del workflow

Infine, scriveremo del codice per avviare l'esecuzione del flusso di lavoro.

1. Apri l'editor di testo e crea il file`WorkflowStarter.java`, aggiungendo una dichiarazione del pacchetto e le importazioni secondo i passaggi comuni.
2. Aggiungi la `WorkflowStarter` classe:

```
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() + "'.");
    }
}
```

La `WorkflowStarter` classe è costituita da un unico metodo`main`, che accetta un argomento opzionale passato dalla riga di comando come dati di input per il flusso di lavoro.

Il metodo client SWF`startWorkflowExecution`, accetta un StartWorkflowExecutionRequestoggetto come input. Qui, oltre a specificare il dominio e il tipo di workflow da eseguire, forniamo:

- un nome di esecuzione del flusso di lavoro leggibile dall'uomo
- dati di input del flusso di lavoro (forniti nella riga di comando nel nostro esempio)
- un valore di timeout che rappresenta il tempo, in secondi, necessario per l'esecuzione dell'intero flusso di lavoro.

L'oggetto Run che `startWorkflowExecution` restituisce fornisce un ID di esecuzione, un valore che può essere utilizzato per identificare questa particolare esecuzione Amazon SWF del flusso di lavoro nella cronologia delle esecuzioni del flusso di lavoro.

+ NOTA: L'ID di esecuzione viene generato da Amazon SWF e non è lo stesso nome di esecuzione del flusso di lavoro passato all'avvio dell'esecuzione del flusso di lavoro.

Crea l'esempio

Per creare il progetto di esempio con Maven, vai alla `helloswf` directory e digita:

```
mvn package
```

Il risultato `helloswf-1.0.jar` verrà generato nella directory. `target`

Esegui l'esempio

L'esempio è costituito da quattro classi eseguibili separate, che vengono eseguite indipendentemente l'una dall'altra.

> **Note**
>
> Se utilizzi un sistema Linux, macOS o Unix, puoi eseguirli tutti, uno dopo l'altro, in un'unica finestra di terminale. Se utilizzi Windows, dovresti aprire due istanze da riga di comando aggiuntive e accedere alla directory di ciascuna. `helloswf`

Impostazione del percorso di classe Java

Sebbene Maven abbia gestito le dipendenze per te, per eseguire l'esempio, dovrai fornire la libreria AWS SDK e le relative dipendenze sul tuo classpath Java. Puoi impostare la variabile di `CLASSPATH` ambiente sulla posizione delle librerie SDK e sulla `third-party/lib` directory nell' AWS SDK, che include le dipendenze necessarie:

```
export CLASSPATH='target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/
lib/*'
java example.swf.hello.HelloTypes
```

oppure usa l'`-cp`opzione del **`java`** comando per impostare il classpath durante l'esecuzione di ciascuna applicazione.

```
java -cp target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/* \
  example.swf.hello.HelloTypes
```

Lo stile che usi dipende da te. Se non avete avuto problemi a compilare il codice, provate entrambi a eseguire gli esempi e ottenete una serie di errori "NoClassDefFound", probabilmente perché il classpath è impostato in modo errato.

Registra il dominio, il flusso di lavoro e i tipi di attività

Prima di avviare i worker e il workflow starter, dovrai registrare il dominio e i tipi di flusso di lavoro e attività. Il codice per eseguire questa operazione è stato implementato in Registra un dominio, flussi di lavoro e tipi di attività.

Dopo la creazione, e se hai impostato CLASSPATH, puoi eseguire il codice di registrazione eseguendo il comando:

```
    echo 'Supply the name of one of the example classes as an argument.'
```

Avvia l'attività e i lavoratori del flusso di lavoro

Ora che i tipi sono stati registrati, puoi avviare i lavoratori dell'attività e del flusso di lavoro. Queste continueranno a funzionare e a verificare le attività finché non verranno interrotte, quindi dovresti eseguirle in finestre di terminale separate oppure, se utilizzi Linux, macOS o Unix, puoi usare & l'operatore per far sì che ognuna di esse generi un processo separato durante l'esecuzione.

```
    echo 'If there are arguments to the class, put them in quotes after the class
name.'
    exit 1
```

Se esegui questi comandi in finestre separate, ometti l'operatore finale & da ogni riga.

Avviate l'esecuzione del flusso di lavoro

Ora che gli addetti alle attività e al flusso di lavoro stanno effettuando i sondaggi, puoi iniziare l'esecuzione del flusso di lavoro. Questo processo verrà eseguito fino a quando il flusso di lavoro non tornerà allo stato completato. Dovreste eseguirlo in una nuova finestra di terminale (a meno che non abbiate eseguito i worker come nuovi processi generati utilizzando l'&operatore).

```
fi
```

> **Note**
>
> Se desideri fornire i tuoi dati di input, che verranno passati prima al flusso di lavoro e poi all'attività, aggiungili alla riga di comando. Per esempio:
>
> ```
> echo "## Running $className..."
> ```

Una volta iniziata l'esecuzione del flusso di lavoro, dovresti iniziare a vedere l'output fornito sia dai lavoratori che dall'esecuzione del flusso di lavoro stesso. Quando il flusso di lavoro sarà finalmente completato, il relativo output verrà stampato sullo schermo.

Fonte completa per questo esempio

Puoi sfogliare il codice sorgente completo di questo esempio su Github nel aws-java-developer-guiderepository.

Ulteriori informazioni

- I lavoratori qui presentati possono comportare la perdita di attività se vengono fermati mentre è ancora in corso un sondaggio sul flusso di lavoro. Per scoprire come chiudere i dipendenti in modo corretto, consulta Shutting Down Activity e Workflow Workers Gracefully.
- Per saperne di più Amazon SWF, visita la Amazon SWFhome page o visualizza la Guida per gli sviluppatori.Amazon SWF
- Puoi usare AWS Flow Framework for Java per scrivere flussi di lavoro più complessi in un elegante stile Java usando le annotazioni. Per ulteriori informazioni, consulta la AWS Flow Framework Java Developer Guide.

Lambda Compiti

In alternativa o in combinazione con Amazon SWF le attività, puoi utilizzare le funzioni Lambda per rappresentare le unità di lavoro nei flussi di lavoro e programmarle in modo simile alle attività.

Questo argomento si concentra su come implementare le Amazon SWF Lambda attività utilizzando. AWS SDK per Java Per ulteriori informazioni sulle Lambda attività in generale, consulta AWS Lambda Tasks nella Amazon SWF Developer Guide.

Configura un ruolo IAM multiservizio per eseguire la tua funzione Lambda

Prima di Amazon SWF poter eseguire la tua Lambda funzione, devi configurare un ruolo IAM che Amazon SWF autorizzi l'esecuzione di Lambda funzioni per tuo conto. Per informazioni complete su come eseguire questa operazione, consulta AWS Lambda Attività.

Avrai bisogno dell'Amazon Resource Name (ARN) di questo ruolo IAM quando registri un flusso di lavoro che Lambda utilizzerà le attività.

Crea una funzione Lambda

È possibile scrivere Lambda funzioni in diversi linguaggi, incluso Java. Per informazioni complete su come creare, distribuire e utilizzare Lambda le funzioni, consulta la Guida per gli AWS Lambda sviluppatori.

> **Note**
>
> Indipendentemente dalla lingua utilizzata per scrivere la Lambda funzione, questa può essere pianificata ed eseguita da qualsiasi Amazon SWF flusso di lavoro, indipendentemente dalla lingua in cui è scritto il codice del flusso di lavoro. Amazon SWF gestisce i dettagli relativi all'esecuzione della funzione e al trasferimento dei dati da e verso di essa.

Ecco una semplice Lambda funzione che potrebbe essere utilizzata al posto dell'attività in Building a Simple Amazon SWF Application.

- Questa versione è scritta in JavaScript, e può essere inserita direttamente utilizzando Console di gestione AWS:

```
exports.handler = function(event, context) {
    context.succeed("Hello, " + event.who + "!");
};
```

- Ecco la stessa funzione scritta in Java, che puoi anche distribuire ed eseguire su Lambda:

```
package example.swf.hellolambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.util.json.JSONException;
import com.amazonaws.util.json.JSONObject;

public class SwfHelloLambdaFunction implements RequestHandler<Object, Object> {
    @Override
    public Object handleRequest(Object input, Context context) {
        String who = "{SWF}";
        if (input != null) {
            JSONObject jso = null;
            try {
                jso = new JSONObject(input.toString());
```

```
                who = jso.getString("who");
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        return ("Hello, " + who + "!");
    }
}
```

> **Note**
>
> Per ulteriori informazioni sulla distribuzione delle funzioni Java in Lambda, consulta Creating a Deployment Package (Java) nella AWS Lambda Developer Guide. Ti consigliamo anche di consultare la sezione intitolata Modello di programmazione per le funzioni di creazione in Java. Lambda

Lambda le funzioni accettano un evento o un oggetto di input come primo parametro e un oggetto di contesto come secondo, che fornisce informazioni sulla richiesta di esecuzione della Lambda funzione. Questa particolare funzione prevede che l'input sia in JSON, con un `who` campo impostato sul nome usato per creare il messaggio di saluto.

Registrare un flusso di lavoro da utilizzare con Lambda

Affinché un flusso di lavoro Lambda pianifichi una funzione, devi fornire il nome del ruolo IAM che fornisce l'autorizzazione Amazon SWF a richiamare Lambda le funzioni. È possibile impostarlo durante la registrazione del flusso di lavoro utilizzando i `setDefaultLambdaRole` metodi `withDefaultLambdaRole` o di RegisterWorkflowTypeRequest.

```
System.out.println("** Registering the workflow type '" + WORKFLOW + "-" +
 WORKFLOW_VERSION
        + "'.");
try {
    swf.registerWorkflowType(new RegisterWorkflowTypeRequest()
        .withDomain(DOMAIN)
        .withName(WORKFLOW)
        .withDefaultLambdaRole(lambda_role_arn)
        .withVersion(WORKFLOW_VERSION)
        .withDefaultChildPolicy(ChildPolicy.TERMINATE)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskStartToCloseTimeout("30"));
```

```
}
catch (TypeAlreadyExistsException e) {
```

Pianifica un' Lambda attività

La pianificazione di un' Lambda attività è simile alla pianificazione di un'attività. Fornisci una decisione con un `ScheduleLambdaFunction` DecisionTypee con ScheduleLambdaFunctionDecisionAttributes.

```
running_functions == 0 && scheduled_functions == 0) {
AWSLambda lam = AWSLambdaClientBuilder.defaultClient();
GetFunctionConfigurationResult function_config =
    lam.getFunctionConfiguration(
            new GetFunctionConfigurationRequest()
                .withFunctionName("HelloFunction"));
String function_arn = function_config.getFunctionArn();

ScheduleLambdaFunctionDecisionAttributes attrs =
    new ScheduleLambdaFunctionDecisionAttributes()
        .withId("HelloFunction (Lambda task example)")
        .withName(function_arn)
        .withInput(workflow_input);

decisions.add(
```

In`ScheduleLambdaFuntionDecisionAttributes`, è necessario fornire un nome, che è l'ARN della Lambda funzione da chiamare, e un id, che è il nome che Amazon SWF verrà utilizzato per identificare la Lambda funzione nei registri cronologici.

È inoltre possibile fornire un input opzionale per la Lambda funzione e impostarne il valore di timeout di inizio e chiusura, ovvero il numero di secondi in cui la Lambda funzione può essere eseguita prima della generazione di un evento. `LambdaFunctionTimedOut`

> **Note**
>
> Questo codice utilizza il AWSLambdaClient per recuperare l'ARN della funzione, dato Lambda il nome della funzione. Puoi usare questa tecnica per evitare di codificare l'ARN completo (che include il tuo Account AWS ID) nel codice.

Gestisci gli eventi della funzione Lambda nel tuo decisore

Lambda le attività genereranno una serie di eventi sui quali è possibile intervenire durante la selezione delle attività decisionali del proprio addetto al flusso di lavoro, corrispondenti al ciclo di vita dell' Lambda attività, con EventTypevalori come, e. `LambdaFunctionScheduled` `LambdaFunctionStarted` `LambdaFunctionCompleted` Se la Lambda funzione fallisce o l'esecuzione impiega più tempo rispetto al valore di timeout impostato, riceverai rispettivamente un tipo di `LambdaFunctionTimedOut` evento `LambdaFunctionFailed` o un tipo di evento.

```
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;
```

Ricevi l'output dalla tua funzione Lambda

Quando ricevete un «`LambdaFunctionCompleted`EventType, you can retrieve your 0 function's return value by first calling `getLambdaFunctionCompletedEventAttributeson» HistoryEventper ottenere un LambdaFunctionCompletedEventAttributesoggetto e poi chiamate il relativo `getResult` metodo per recuperare l'output della Lambda funzione:

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

Fonte completa per questo esempio

Puoi sfogliare la fonte completa:github: `< awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf _lambda/> per questo esempio su Github nel repository. aws-java-developer-guide

Chiusura graduale degli addetti alle attività e ai flussi di lavoro

L'argomento Creazione di un' Amazon SWF applicazione semplice ha fornito un'implementazione completa di una semplice applicazione per il flusso di lavoro composta da un'applicazione di registrazione, un addetto alle attività e ai flussi di lavoro e uno starter del flusso di lavoro.

Le classi di lavoro sono progettate per essere eseguite in modo continuo e consentono di eseguire sondaggi sulle attività inviate Amazon SWF per eseguire attività o restituire decisioni. Una volta effettuata una richiesta di sondaggio, Amazon SWF registra il poller e tenterà di assegnargli un compito.

Se il lavoratore del flusso di lavoro viene licenziato durante un sondaggio prolungato, Amazon SWF può comunque provare a inviare un'attività al lavoratore terminato, con conseguente perdita dell'attività (fino al timeout dell'attività).

Un modo per gestire questa situazione consiste nell'attendere la restituzione di tutte le lunghe richieste di sondaggio prima che il lavoratore termini.

In questo argomento, riscriveremo l'Activity Worker utilizzando gli hook di `helloswf` shutdown di Java per tentare di arrestare correttamente l'activity worker.

Ecco il codice completo:

```
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 questa versione, il codice di polling presente nella `main` funzione nella versione originale è stato spostato nel suo metodo,`pollAndExecute`.

La `main` funzione ora utilizza CountDownLatcha insieme a un hook di spegnimento per far attendere il thread fino a 60 secondi dopo la richiesta di terminazione prima di chiudere il thread.

Registrazione di domini

Ogni flusso di lavoro e attività Amazon SWFrichiede un dominio in cui essere eseguito.

1. Crea un nuovo RegisterDomainRequestoggetto, fornendogli almeno il nome di dominio e il periodo di conservazione dell'esecuzione del flusso di lavoro (entrambi i parametri sono obbligatori).
2. Chiamate il metodo AmazonSimpleWorkflowClient.registerDomain con l'oggetto. RegisterDomainRequest
3. Verifica DomainAlreadyExistsExceptionse il dominio che stai richiedendo esiste già (nel qual caso, di solito non è richiesta alcuna azione).

Il codice seguente illustra questa procedura:

```
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!");
    }
```

```
}
```

Elenco di domini

Puoi elencare i Amazon SWFdomini associati al tuo account e alla tua AWS regione in base al tipo di registrazione.

1. Crea un ListDomainsRequestoggetto e specifica lo stato di registrazione dei domini che ti interessano (questo è obbligatorio).
2. Chiama AmazonSimpleWorkflowClient.listDomains con l'oggetto. ListDomainRequest I risultati vengono forniti in un oggetto. DomainInfos
3. Richiama getDomainInfosl'oggetto restituito per ottenere un elenco di DomainInfooggetti.
4. Chiama getName su ogni DomainInfooggetto per ottenerne il nome.

Il codice seguente illustra questa procedura:

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

Esempi di codice inclusi nell'SDK

AWS SDK per Java Viene fornito con esempi di codice che illustrano molte delle funzionalità dell'SDK in programmi compilabili ed eseguibili. È possibile studiarli o modificarli per implementare le proprie soluzioni utilizzando il. AWS AWS SDK per Java

Come ottenere i campioni

Gli esempi di AWS SDK per Java codice sono forniti nella directory samples dell'SDK. Se hai scaricato e installato l'SDK utilizzando le informazioni in Configurazione AWS SDK per Java, gli esempi sono già presenti sul tuo sistema.

Puoi anche visualizzare gli esempi più recenti nel AWS SDK per Java GitHub repository, nella directory src/samples.

Creazione ed esecuzione degli esempi utilizzando la riga di comando

Gli esempi includono gli script di compilazione di Ant in modo da poterli creare ed eseguire facilmente dalla riga di comando. Ogni esempio contiene anche un file README in formato HTML che contiene informazioni specifiche per ogni esempio.

> **Note**
>
> Se stai sfogliando il codice di esempio GitHub, fai clic sul pulsante Raw nella visualizzazione del codice sorgente quando visualizzi il file README.html dell'esempio. In modalità raw, il codice HTML verrà visualizzato come previsto nel browser.

Prerequisiti

Prima di eseguire qualsiasi esempio, AWS SDK per Java è necessario impostare le AWS credenziali nell'ambiente o con AWS CLI, come specificato in Configurazione delle AWS credenziali e della regione per lo sviluppo. Gli esempi utilizzano la catena di fornitori di credenziali predefinita quando possibile. Quindi, impostando le credenziali in questo modo, è possibile evitare la pratica rischiosa di inserire AWS le credenziali in file all'interno della directory del codice sorgente (dove potrebbero essere inavvertitamente archiviate e condivise pubblicamente).

Esecuzione degli esempi

1. Passate alla directory contenente il codice dell'esempio. Ad esempio, se ti trovi nella directory principale del download dell' AWS SDK e desideri eseguire l'`AwsConsoleApp`esempio, devi digitare:

```
cd samples/AwsConsoleApp
```

2. Compila ed esegui l'esempio con Ant. Il target di build predefinito esegue entrambe le azioni, quindi puoi semplicemente inserire:

```
ant
```

L'esempio stampa le informazioni sullo standard output, ad esempio:

```
===========================================

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.
```

Creazione ed esecuzione degli esempi utilizzando l'IDE Eclipse

Se utilizzi il AWS Toolkit for Eclipse, puoi anche avviare un nuovo progetto in Eclipse basato su AWS SDK per Java o aggiungere l'SDK a un progetto Java esistente.

Prerequisiti

Dopo aver installato AWS Toolkit for Eclipse, ti consigliamo di configurare il Toolkit con le tue credenziali di sicurezza. Puoi farlo in qualsiasi momento scegliendo Preferenze dal menu Finestra di Eclipse, quindi scegliendo la sezione Toolkit. AWS

Esecuzione degli esempi

1. Aprire Eclipse
2. Crea un nuovo progetto Java AWS . In Eclipse, nel menu File, scegliete Nuovo, quindi fate clic su Progetto. Si apre la procedura guidata Nuovo progetto.
3. Espandi la AWS categoria, quindi scegli AWS Java Project.
4. Scegli Next (Successivo). Viene visualizzata la pagina delle impostazioni del progetto.

5. Immettete un nome nella casella Nome progetto. Il gruppo AWS SDK per Java Samples mostra gli esempi disponibili nell'SDK, come descritto in precedenza.
6. Seleziona gli esempi che desideri includere nel progetto selezionando ciascuna casella di controllo.
7. Inserisci le tue AWS credenziali. Se le hai già configurate AWS Toolkit for Eclipse con le tue credenziali, queste vengono compilate automaticamente.
8. Scegli Fine. Il progetto viene creato e aggiunto a Project Explorer.
9. Scegli il `.java` file di esempio che desideri eseguire. Ad esempio, per l' Amazon S3 esempio, scegli`S3Sample.java`.
10. Scegliete Esegui dal menu Esegui.
11. Fate clic con il pulsante destro del mouse sul progetto in Project Explorer, scegliete Crea percorso, quindi scegliete Aggiungi librerie.
12. Scegliete AWS Java SDK, scegliete Avanti, quindi seguite le restanti istruzioni visualizzate sullo schermo.

Sicurezza per AWS SDK per Java

La sicurezza cloud di Amazon Web Services (AWS) è la priorità più alta. In qualità di cliente AWS , è possibile trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle organizzazioni più esigenti a livello di sicurezza. La sicurezza è una responsabilità condivisa tra te e te. AWS Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud.

Security of the Cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce tutti i servizi offerti nel AWS Cloud e della fornitura di servizi che è possibile utilizzare in modo sicuro. La nostra responsabilità in AWS materia di sicurezza è la massima priorità e l'efficacia della nostra sicurezza viene regolarmente testata e verificata da revisori di terze parti nell'ambito dei Programmi di AWS conformità.

Sicurezza nel cloud: la responsabilità dell'utente è determinata dal AWS servizio utilizzato e da altri fattori, tra cui la sensibilità dei dati, i requisiti dell'organizzazione e le leggi e i regolamenti applicabili.

Questo AWS prodotto o servizio segue il modello di responsabilità condivisa attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la pagina della documentazione sulla sicurezza del AWS servizio e AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Argomenti

Protezione dei dati in AWS SDK per Java 1.x

Il modello di responsabilità condivisa si applica alla protezione dei dati in questo AWS prodotto o servizio. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura

globale che gestisce tutto il AWS cloud. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Questo contenuto include la configurazione della protezione e le attività di gestione per i servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, consulta Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il modello di responsabilitàAWS condivisa e il post sul blog sul GDPR sul AWS Security Blog.

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e di configurare account utente individuali con AWS Identity and Access Management (IAM). In questo modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere il proprio lavoro. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Utilizzatelo per comunicare con AWS le risorse.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizza soluzioni di AWS crittografia, con tutti i controlli di sicurezza predefiniti all'interno AWS dei servizi.
- Utilizza servizi di sicurezza gestiti avanzati come Amazon Macie, che aiuta a scoprire e proteggere i dati personali archiviati in. Amazon S3
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Consigliamo di non inserire mai informazioni identificative sensibili, ad esempio i numeri di account dei clienti, in campi a formato libero come un campo Nome. Ciò include quando si utilizza questo AWS prodotto o servizio o altri AWS servizi utilizzando la console, l'API o. AWS CLI AWS SDKs Tutti i dati inseriti in questo AWS prodotto o servizio o in altri servizi potrebbero essere raccolti per essere inclusi nei registri di diagnostica. Quando fornisci un URL a un server esterno, non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta a tale server.

AWS SDK per Java supporto per TLS

Le seguenti informazioni si applicano solo all'implementazione Java SSL (l'implementazione SSL predefinita in). AWS SDK per Java Se usi un'implementazione SSL diversa, vedi l'implementazione SSL specifica per informazioni su come applicare le versioni TLS.

Come controllare la versione di TLS

Consultate la documentazione del provider della macchina virtuale Java (JVM) per determinare quali versioni TLS sono supportate sulla vostra piattaforma. Per alcuni JVMs, il codice seguente stamperà quali versioni SSL sono supportate.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProto
```

Per vedere l'handshake SSL in azione e quale versione di TLS viene utilizzata, puoi utilizzare la proprietà di sistema javax.net.debug.

```
java app.jar -Djavax.net.debug=ssl
```

> **Note**
>
> TLS 1.3 non è compatibile con le versioni SDK for Java da 1.9.5 a 1.10.31. Per ulteriori informazioni, consulta il seguente post di blog. https://aws.amazon.com/blogs/developer/tls-1-3- - -1-9-5-to-1-10-31/ incompatibility-with-aws-sdk for-java-versions

Applicazione di una versione minima di TLS

L'SDK preferisce sempre l'ultima versione TLS supportata dalla piattaforma e dal servizio. Se desideri applicare una versione TLS minima specifica, consulta la documentazione della tua JVM. Per i sistemi basati su OpenJDK JVMs, è possibile utilizzare la proprietà di sistema. `jdk.tls.client.protocols`

```
java app.jar -Djdk.tls.client.protocols=PROTOCOLS
```

Consultate la documentazione della vostra JVM per i valori supportati dei PROTOCOLS.

Identity and Access Management

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori

IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse. AWS IAM è uno Servizio AWS strumento che puoi utilizzare senza costi aggiuntivi.

Argomenti

Destinatari

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro che AWS svolgi.

Utente del servizio: se lo utilizzi Servizi AWS per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più AWS funzionalità per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell’accesso consente di richiedere le autorizzazioni corrette all’amministratore. Se non riesci ad accedere a una funzionalità di AWS, consulta Risoluzione dei problemi di AWS identità e accesso o consulta la guida per l'utente della funzionalità Servizio AWS che stai utilizzando.

Amministratore del servizio: se sei responsabile delle AWS risorse della tua azienda, probabilmente hai pieno accesso a AWS. È tuo compito determinare a quali AWS funzionalità e risorse devono accedere gli utenti del servizio. È quindi necessario inviare le richieste all’amministratore IAM per modificare le autorizzazioni degli utenti del servizio. Esaminare le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per saperne di più su come la tua azienda può utilizzare IAM con AWS, consulta la guida per l'utente del Servizio AWS software che stai utilizzando.

Amministratore IAM: un amministratore IAM potrebbe essere interessato a ottenere dei dettagli su come scrivere policy per gestire l'accesso a AWS. Per visualizzare esempi di policy AWS basate sull'identità che puoi utilizzare in IAM, consulta la guida per l'utente di quella Servizio AWS che stai utilizzando.

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/ Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere al tuo nella Guida per l'utente Account AWS.Accedi ad AWS

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella IAM User Guide.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Ti consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali dell'utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come best practice, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo. AWS IAM Identity Center Per ulteriori informazioni, consulta Cos'è IAM Identity Center? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è un'identità con autorizzazioni specifiche per una singola persona o applicazione. Consigliamo di utilizzare credenziali temporanee anziché utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un

provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per grandi gruppi di utenti. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente IAM.

I ruoli IAM sono utili per l'accesso federato degli utenti, le autorizzazioni utente IAM temporanee, l'accesso tra account, l'accesso tra servizi e le applicazioni in esecuzione su Amazon. EC2 Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

Gestione dell'accesso con l'utilizzo delle policy

Puoi controllare l'accesso AWS creando policy e collegandole a identità o risorse. AWS Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sui documenti relativi alle policy JSON, consulta Overview of JSON policy nella IAM User Guide.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguire l'operazione.

Policy basate sull'identità

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che alleghi a un'identità (utente, gruppo o ruolo). Queste politiche controllano quali azioni possono eseguire le identità, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su

identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente di IAM.

Le politiche basate sull'identità possono essere politiche in linea (incorporate direttamente in una singola identità) o politiche gestite (politiche autonome collegate a più identità). Per scoprire come scegliere tra politiche gestite e politiche in linea, consulta Choose between managed policy e inline policy nella IAM User Guide.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Le policy basate su risorse sono policy inline che risiedono in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Liste di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta le autorizzazioni massime che una policy basata sull'identità può concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM .
- Politiche di controllo del servizio (SCPs): specifica le autorizzazioni massime per un'organizzazione o un'unità organizzativa in. AWS Organizations Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .

- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta Politiche di controllo delle risorse (RCPs) nella Guida per l'AWS Organizations utente.
- Criteri di sessione: criteri avanzati passati come parametro durante la creazione di una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per sapere come si AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta la logica di valutazione delle policy nella IAM User Guide.

Come Servizi AWS lavorare con IAM

Per avere una visione di alto livello di come Servizi AWS funziona la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Per scoprire come utilizzare uno specifico Servizio AWS con IAM, consulta la sezione sulla sicurezza della Guida per l'utente del servizio pertinente.

Risoluzione dei problemi di AWS identità e accesso

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con un AWS IAM.

Argomenti

- Non sono autorizzato a eseguire alcuna azione in AWS
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire a persone esterne a me di accedere Account AWS alle mie AWS risorse

Non sono autorizzato a eseguire alcuna azione in AWS

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `awes:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `awes:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore indicante che non sei autorizzato a eseguire l'azione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo ad AWS.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in AWS. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

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

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ha fornito le credenziali di accesso.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie AWS risorse

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per

i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per maggiori informazioni, consultare gli argomenti seguenti:

- Per sapere se AWS supporta queste funzionalità, consulta. Come Servizi AWS lavorare con IAM
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consultare Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

Convalida della conformità per questo AWS prodotto o servizio

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità di conformità durante l'utilizzo Servizi AWS, consulta la Documentazione AWS sulla sicurezza.

Questo AWS prodotto o servizio segue il modello di responsabilità condivisa attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la pagina della documentazione sulla sicurezza del AWS servizio e AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Resilienza per questo AWS prodotto o servizio

L'infrastruttura AWS globale è costruita attorno a zone Regioni AWS di disponibilità.

Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti.

Con le zone di disponibilità è possibile progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, vedere Global Infrastructure.AWS

Questo AWS prodotto o servizio segue il modello di responsabilità condivisa attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la pagina della documentazione sulla sicurezza del AWS servizio e AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Sicurezza dell'infrastruttura per questo AWS prodotto o servizio

Questo AWS prodotto o servizio utilizza servizi gestiti ed è pertanto protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzate chiamate API AWS pubblicate per accedere a questo AWS Prodotto o Servizio attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa è possibile utilizzare AWS Security

Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Questo AWS prodotto o servizio segue il modello di responsabilità condivisa attraverso i servizi specifici di Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la pagina della documentazione sulla sicurezza del AWS servizio e AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Amazon S3 Migrazione dei client di crittografia

Questo argomento mostra come migrare le applicazioni dalla versione 1 (V1) del client di crittografia () alla versione 2 Amazon Simple Storage Service (V2 Amazon S3) e garantire la disponibilità delle applicazioni durante tutto il processo di migrazione.

Prerequisiti

Amazon S3 la crittografia lato client richiede quanto segue:

- Java 8 o versione successiva installata nell'ambiente applicativo. AWS SDK per Java Funziona con l'Oracle Java SE Development Kit e con le distribuzioni di Open Java Development Kit (OpenJDK) come Amazon CorrettoRed Hat OpenJDK e JDK. AdoptOpen
- Il pacchetto Bouncy Castle Crypto. Puoi inserire il file.jar di Bouncy Castle nel classpath del tuo ambiente applicativo o aggiungere una dipendenza dall'artifactID `bcprov-ext-jdk15on` (con groupid di) al tuo file Maven. `org.bouncycastle` `pom.xml`

Panoramica sulla migrazione

Questa migrazione avviene in due fasi:

1. Aggiorna i client esistenti per leggere nuovi formati. Aggiorna l'applicazione per utilizzare la versione 1.11.837 o successiva di AWS SDK per Java e ridistribuisci l'applicazione. Ciò consente ai Amazon S3 client del servizio di crittografia lato client dell'applicazione di decrittografare gli oggetti creati dai client del servizio V2. Se l'applicazione ne utilizza più AWS SDKs, è necessario aggiornare ogni SDK separatamente.
2. Migra i client di crittografia e decrittografia alla V2. Una volta che tutti i client di crittografia V1 saranno in grado di leggere i formati di crittografia V2, aggiorna i Amazon S3 client di crittografia e decrittografia lato client nel codice dell'applicazione per utilizzare i loro equivalenti V2.

Aggiorna i client esistenti per leggere nuovi formati

Il client di crittografia V2 utilizza algoritmi di crittografia che le versioni precedenti di AWS SDK per Java non supportano.

Il primo passaggio della migrazione consiste nell'aggiornare i client di crittografia V1 per utilizzare la versione 1.11.837 o successiva di. AWS SDK per Java(Ti consigliamo di eseguire l'aggiornamento alla versione più recente, che puoi trovare nella versione Java API Reference 1.x.) A tale scopo, aggiorna la dipendenza nella configurazione del progetto. Dopo aver aggiornato la configurazione del progetto, ricostruisci il progetto e ridistribuiscilo.

Una volta completati questi passaggi, i client di crittografia V1 dell'applicazione saranno in grado di leggere gli oggetti scritti dai client di crittografia V2.

Aggiorna la dipendenza nella configurazione del tuo progetto

Modifica il file di configurazione del progetto (ad esempio, pom.xml o build.gradle) per utilizzare la versione 1.11.837 o successiva di. AWS SDK per Java Quindi, ricostruisci il progetto e ridistribuiscilo.

Il completamento di questo passaggio prima di implementare il nuovo codice applicativo aiuta a garantire che le operazioni di crittografia e decrittografia rimangano coerenti in tutta la flotta durante il processo di migrazione.

Esempio di utilizzo di Maven

Frammento di codice da un file pom.xml:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.11.837</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Esempio di utilizzo di Gradle

Frammento di codice da un file build.gradle:

```
dependencies {
  implementation platform('com.amazonaws:aws-java-sdk-bom:1.11.837')
  implementation 'com.amazonaws:aws-java-sdk-s3'
}
```

Migrazione dei client di crittografia e decrittografia alla V2

Una volta aggiornato il progetto con l'ultima versione SDK, puoi modificare il codice dell'applicazione per utilizzare il client V2. A tale scopo, per prima cosa aggiorna il codice per utilizzare il nuovo service client builder. Quindi fornisci materiali di crittografia utilizzando un metodo sul builder che è stato rinominato e configura ulteriormente il client di servizio secondo necessità.

Questi frammenti di codice dimostrano come utilizzare la crittografia lato client con e forniscono confronti tra i AWS SDK per Java client di crittografia V1 e 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();
```

L'esempio precedente imposta `cryptoMode` su`AuthenticatedEncryption`. Questa è un'impostazione che consente a un client di crittografia V2 di leggere oggetti scritti da un client di crittografia V1. Se il tuo client non ha bisogno della capacità di leggere oggetti scritti da un client V1, ti consigliamo di utilizzare invece l'impostazione predefinita di. `StrictAuthenticatedEncryption`

Costruisci un client di crittografia V2

Il client di crittografia V2 può essere creato chiamando AmazonS3 EncryptionClient v2.encryptionBuilder ().

Puoi sostituire tutti i client di crittografia V1 esistenti con client di crittografia V2. Un client di crittografia V2 sarà sempre in grado di leggere qualsiasi oggetto che è stato scritto da un client di crittografia V1 purché gli si consenta di farlo configurando il client di crittografia V2 per utilizzare il `. AuthenticatedEncryption `cryptoMode

La creazione di un nuovo client di crittografia V2 è molto simile a come si crea un client di crittografia V1. Tuttavia, ci sono alcune differenze:

- Utilizzerai un `CryptoConfigurationV2` oggetto per configurare il client anziché un `CryptoConfiguration` oggetto. Questo parametro è obbligatorio.
- L'`cryptoMode`impostazione predefinita per il client di crittografia V2 è`StrictAuthenticatedEncryption`. Per il client di crittografia V1 lo è. `EncryptionOnly`
- Il metodo withEncryptionMaterials() sul generatore del client di crittografia è stato rinominato withEncryptionMaterialsProvider (). Si tratta semplicemente di una modifica estetica che riflette in modo più accurato il tipo di argomento. È necessario utilizzare il nuovo metodo quando si configura il client di servizio.

> **Note**
>
> Quando decifrate con AES-GCM, leggete l'intero oggetto fino alla fine prima di iniziare a utilizzare i dati decrittografati. Questo serve a verificare che l'oggetto non sia stato modificato da quando è stato crittografato.

Utilizza fornitori di materiali di crittografia

Puoi continuare a utilizzare gli stessi fornitori di materiali di crittografia e gli stessi oggetti di materiale di crittografia che stai già utilizzando con il client di crittografia V1. Queste classi hanno la responsabilità di fornire le chiavi utilizzate dal client di crittografia per proteggere i dati. Possono essere utilizzate in modo intercambiabile sia con il client di crittografia V2 che con il client di crittografia V1.

Configurare il client di crittografia V2

Il client di crittografia V2 è configurato con un `CryptoConfigurationV2` oggetto. Questo oggetto può essere costruito chiamando il relativo costruttore predefinito e quindi modificandone le proprietà come richiesto dai valori predefiniti.

I valori predefiniti per sono: `CryptoConfigurationV2`

- `cryptoMode` = `CryptoMode.StrictAuthenticatedEncryption`
- `storageMode` = `CryptoStorageMode.ObjectMetadata`
- `secureRandom`= istanza di `SecureRandom`
- `rangeGetMode` = `CryptoRangeGetMode.DISABLED`
- `unsafeUndecryptableObjectPassthrough` = `false`

Si noti che non EncryptionOnlyè supportato `cryptoMode` nel client di crittografia V2. Il client di crittografia V2 crittograferà sempre i contenuti utilizzando la crittografia autenticata e proteggerà le chiavi di crittografia dei contenuti () CEKs utilizzando oggetti V2. `KeyWrap`

L'esempio seguente mostra come specificare la configurazione di crittografia in V1 e come creare un'istanza di un oggetto V2 da passare al generatore del client di crittografia CryptoConfigurationV2.

V1

```
CryptoConfiguration cryptoConfiguration = new CryptoConfiguration()
        .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

V2

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
        .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

Esempi aggiuntivi

Gli esempi seguenti mostrano come affrontare casi d'uso specifici relativi a una migrazione dalla V1 alla V2.

Configurare un client di servizio per leggere gli oggetti creati dal client di crittografia V1

Per leggere oggetti scritti in precedenza utilizzando un client di crittografia V1, imposta su. `cryptoMode` `AuthenticatedEncryption` Il seguente frammento di codice mostra come costruire un oggetto di configurazione con questa impostazione.

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
        .withCryptoMode(CryptoMode.AuthenticatedEncryption);
```

Configura un client di servizio per ottenere intervalli di byte di oggetti

Per poter recuperare `get` un intervallo di byte da un oggetto S3 crittografato, abilita la nuova impostazione di configurazione. `rangeGetMode` Per impostazione predefinita, questa impostazione è disabilitata sul client di crittografia V2. Nota che, anche se abilitato, un intervallo funziona `get` solo su oggetti che sono stati crittografati utilizzando algoritmi supportati dall'`cryptoMode`impostazione del client. Per ulteriori informazioni, consulta l' AWS SDK per Java API CryptoRangeGetModeReference.

Se intendi utilizzare il per Amazon S3 TransferManager eseguire download in più parti di Amazon S3 oggetti crittografati utilizzando il client di crittografia V2, devi prima abilitare l'`rangeGetMode`impostazione sul client di crittografia V2.

Il seguente frammento di codice mostra come configurare il client V2 per l'esecuzione di un intervallo. `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);
```

Chiave OpenPGP per AWS SDK per Java

Tutti gli artefatti Maven disponibili pubblicamente per il AWS SDK per Java sono firmati utilizzando lo standard OpenPGP. La chiave pubblica necessaria per verificare la firma di un artefatto è disponibile nella sezione seguente.

Chiave attuale

La tabella seguente mostra le informazioni chiave di OpenPGP per le versioni correnti di SDK for Java 1x e SDK for Java 2.x.

ID chiave	0x 07B386692DADD AC1
Tipo	RSA
Size	4096/4096
Creato	30-06-2016
Scade	2026-09-27
ID utente	AWS SDKs e strumenti < @amazon .com> aws-dr-tools
Impronta digitale chiave	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 692 PAPÀ AC1

Per copiare la seguente chiave pubblica OpenPGP per l'SDK for Java negli appunti, seleziona l'icona «Copia» nell'angolo in alto a destra.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Hostname:
Version: Hockeypuck 2.2

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

```
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJo12ZrBQkTQxnmAAoJEKwQezhmktrdi18P/A3De83MBx8bdcWJ
Fot71Vk1TyBQFErgrtcytSU0czEHx3tGbzgQLbMlyzjirOT03usxEkOeqTVK+RU+
5uFXNZYQLwMJlHJ6S8tnfLe/ExM5WQ2KPwIUPfZs1GDDRQB2dIKSc+qYrP1O1vf4
O4iPgfLHMW2bFh3zjjxcaHCJyqc7Cau33eZFBAsRni1jOUo7MeyX0hlXfW8pd48Q
wZllQVZ/6KmDiFWA0CZ+2svJ5cL0tgPoh1OQjoz0nHpNfuDILMrZ+e7tx2VTlkGH
UGeNSydnrK8v9ztFn34KtU/k7NEWoVSyEi+5ICZL18FBwPqTwdVWXwXrqZCKiIpr
8ZdJWDz2sJfgDFNCC6rKgCQ6FrmaD9G76dYWkQ4AbZqABlUzU3q36W1K0r3iOAb5
G4tdOt4yqXHTe1x+ZUNaeW7gaCmtXAxLw0OfeJrcq/44b/SQP+qJ8sSOv76Yg2oF
BsF5DWOVUFghbTyokHAoVROyhBR4dUUisY39AqLSL8+Lp9Pr3wNuGl9GLrMD5701
piUb88B3Gwe1EiKV1gaKrvZ3mECDUiSMVO0Z5iG8E4QDpNmVbJbV1uT821ubvtOv
2Ko10Fa0uwCYGssdRGqEXNy6jz/Er8LAC3+nmGINDJQzrF+loYoSSkI2Nu7lhMuL
7iWwUPF7OhDXoVSAn4X3x6q2rGK0wsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZTsFCRNDGLYACgkQ
rBB7OGaS2t0/Ow//YIv51vHtD+kwMmIvk3zpizDHY0zW2dOezAo+C/DsSyC7wDll
Dixw34EQ1yLXH5xLR8CH1zupl3JmmEp1ucdQggoefbidxDl8Fld7tJOD1y3GGnTD
0jAl2ZC+W65Oh+wS1mD1FlaKjMGGkvJf0dA7RtU2T8dv3vt8dsxg76FMFS3+fqlC
FNOAsNTn9zWR1SqBIfkMJK83aq6s/rcEV9VrAYgDgqex58fygB5EuTf842/IF7WZ
Q9gd6fupB0mMZP5YWd2uj/vsBTYakG+mgQwDxZuKPeEzAqnqqS7biSQOUO6Wozlq
Yy4fSczE9GkBAvg0pGmbko+zHvpnjvX/h1CUpC6odvFyOAhZp6zyhs0QWz9thfqV
lU8WlbgJ2atFDn5GUSxF/fe0Yzovlbbs6sbYXuvMG9RiEOuJ1mBbZR3aIdZ1U6Do
BHc/vjc5mWcV7JQSP7i4W/8W7X3UAuN9LdxB+IvF3Cwrgtlw2BWvA5Alco5Tnz8t
P/CIVmBjk+sLme8W4kfLK3IWEbwClOdNnErI/MHRm65A2Y5EMIhwjrOi07SU1Pxa
nPpg3OYJCdvjzdB8QE3/DBiMfOl4dISfKDVEWnfK8mZaYd/BeRm2gUAa9UrqSFCG
BlA7Lg+eLI3US0FvwWJ4j5bBJqgLu+y7crIkiUOPAQuLk3lO+5uYU/I3DuLCwZQE
EwEKAD4CGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AWIQT+uSCfLy8/RmSEHlWs
EHs4ZpLa3QUCZwAXCAUJEWvKgwAKCRCsEHs4ZpLa3ZdTEACMBLg2q9zk8ZH02nDz
Sg5zc8Wlqq8WdxU0Pj8qx4UOrrMca7wyiUvrgoxPW5lh1RVNUeMkDRfu9pSXcOVI
V9LvmYE/WnwKROubgGbsC4T7M/LqV0/AulXil4d7IXcO6l4toa8LTNWtD5bODgrN
gvay1AzCU8kq1Qw1cKZ2gAfvA3Ba7PWyLeUN4HTlGrXcw73G+0CofY1L8wqWxHCJ
29XqQzeTEc6MDEeIlNlVdUcy8Qr5uwkEsl34H9AxS5F1opJ4TqvXiDZsrSRRv57R
XYmRZDWeYT+9PZaMsHXza5qgej7BfATxhYfICsNaY6MK3x6b+nDSKkoZgO+i09zh
1YjpahhQe6G336v/3mRj0dKGCRQ6znQ9ghUaB5z9zfvgH5AOEkTe3l8MqM+j5A6P
VjSBBJAHKejxr7+wKJKIA6P+DqpsYAunzftwUzrLVqb+BZQ+DcTmVrE7OPcMYJD5
QglX/Le+WmWZHI154NXgpWWUOUgZUbUge4DKrT+zCJ9iecPLKTW7OcULyXO+rjb8
8BGrD5GPlHB3dOUXXTlMKCqg3qy1Bu2KnZTQiaEEdZgSIGQbrW0JTMmmXJkKjokd
JMA4vYeg5en51G9nRQjScPngx77IxvByNyFWTJdG1ENpJpsK9TtmENcpyUJtJZTJ
```

```
ZSOIRVPP5RzR5vInuXWq6VV0BMLBlAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMB
AAIeAQIXgBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEoiBQkPj/2dAAoJEKwQ
ezhmktrdx1YP/0vvym3jgX/pwnR7K1rafZMb1iKQBr0ISG8cdbaf4pqX5vuUZnyj
w9Cl/oONn7jJjnQxOIIzuBoxne2WN28ftM2w0nVXm85mAmz2fwQz/fdKDyonXcOh
pfD2iMqn7gESjhEgRE7wMDYMDuLdqHI7OKWGVfgrh7xEmKapLh45h7cnumo2VjL9
uDYY1aOBHz993T7oE41y43rhk+6kKbGFd2uuo7h5j1ZF8Lj6sYfcEzXOU1OhRlD0
nyBjDy9MYWu0YNouc70WgMceGx6hjvCAM/5fxP7SZFecZ7ePeB0GpvVA24hSNENE
0r3tUekuOf1I0FunMnMnbh7ZO9rPYqWvWDNIpU3S4CjFhY82L+IeKnmLy8N6ASRk
HsPiNCOHSK8C/0ynrd9xLhX8Jsk/TGiQYaleoHhWkNLlZsL86QHL8SKEqkqZCQf5
AEqghDP6NEGS7lnOenA7JjIrA9KLlT7fnNWZOwFi5X+o/CymE2ytEMS0Yf3nmY4U
n9x56Wgn6J2zqB5nqOXf6NxGdAIgOBm098YEnKCIFzk+yhoDlprVpHcnd2b5f6Oq
uh8KYOEbKgpMJ3zZuWSL5kwGF1nNoYiAkonMaz9H3pOQnOMVYCUeUTDRsiO/prrd
UhNlry4TAsBMpeXnFhdLVM3vFQZVpByadGOJNmnaN/Wavw2a00UGBFa4wsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJhMqGaBQkLnlUVAAoJEKwQ
ezhmktrd2sQP/3YHM+U+BbOy1nSEAfykZ71+uCM2hkHMLdxQYWB/rBWkmg/pbu+d
r4t45RsTASrNjRcZOntlPMQRIq973ymHfpmeS+noFwvTGH7zDv1BRBR9wPrd1XUz
iSuEUHGi/fqxUVXQ5mbonzfThX8tuXeuiQmeToqoB00FYlZm6xsNnEHcjVl66mC4
IPoJLWnZJs4rOCeoRf5XvDTgX6xt5/kLYRZf79qaWGFvaZpsc1CH+rQJUdVa/D4T
7pI7hX6zy0S91z4iuC5HZUiOTF+y5auEZHGTdTWNS1kvOvfcCTi0XK/GkGL82SZu
7X2VGnpCeUnFyViRGlk+KaDG1sVyDY+lcBPg6ilr45M6MQV0iHS5OF04QNXSKt5+
UnzJH7lldgNsR6ibRMyNV3k5v3fyUcSBvIYyLORTTBiVEjQDSbk1QNqbrQlX9CWz
+EJWn16BFTmMFvxBSWPm640GncHP5J3/0MbMw3Cm90x7k8UfNANIemcrJrSxIDwm
g9cVAg3a+D+wxjrVe8jGg0ejvECpm+0yswigj5x6Lqj09A4UgdjEauN+/pn0nhBo
Gv7DzMXtM/LoDtgp6wn93qZVN2TsuHnkEk4UyntB6eWJbBdXHWUr47exiWh0dvQN
tpwCWPT6I7ZTPtA5K/zx+q9m6797BLgAkTYc6gloQL3vs1Z1S3m/hZNawsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJgmrz2BQkLBnBxAAoJEKwQ
ezhmktrd36oP/2rB2EkwSOCKC4m0heWSfDWi6OBKoEbbDtFtc6/HwqBW8SPsiKlq
zV0e3qBY/LVju04+ktJEK+EGXLnC3iC36MegrQ8zt391kEx/Zv9LIuVOCX90QIAX
dL8MVUkkjRLCFFH8pTgRy1cJYWk1X4dYdXWYc29fCwNVartNdNBhsb2ht3VJeKDE
kUivBHmkjuISDPEnI1coY7Lj0ZtY5cHdRF2eZpB0RkTBpsIt18rCYyHkERZrhmvb
j3rOyPyvOa+1/dQS8/hv5pEmbKx8cy8RdJkmbUHYatPBsjHkJSWr7O7G9VFW4GoN
9CRAI4KkbDSEDjCL5dv2pq0Sew1MkLuWJGULAMgiIUlWcOs5SZZGFSksNQrtSFV9
Z/wGocecMGkGQNXQ06JV/Fry/TvyphBlmylEqL+NLqEcEjnlz90IVu+ZA+M09J96
UlHO7V5GvBgM+QK/q/dJeMHPWrNlo1gA6Nwl/HBdM0DqzdZ2jEPvsQSABvZrPMty
+BAqEar4wqY1AH4X5ccEjO7nJQoBQSDRSki1fkBsc1nx44N/m0kHdIa0Z/Y+Mw4v
WiZhREkOospG1I4lBa3CNTVAhSs9msGsYfkqvFJGHL7sZY8XSv82GBBvA0nUNrsJ
bLBwo2FaQG9eoatRAGkqp4b/OtNtBuGeiQoNwFGbfUZTAaStj5/zZj0swsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJe+9bwBQkJZ4plAAoJEKwQ
ezhmktrd+ScP/RoaUKriVVAgLHOGs+/mnfKtnfTlClzi5dsdI9/6H0vLpmSWK/Cl
2cT6gary45VMgAeVK+H1lQXafYj+FY++I5kYoe2GrSvIXhpjaFAJyNf/dKleTsqR
Tm371i8b3FDYs5kvy2CnTbmHB8MsOGxck8/YHd1x+g8WpO2IgF89yYCSF3CAdxC3
6bHbs6Z3C3lcM/3SoWF+Yie2P8XeBMPCGp/BcjQzUcHF6G06TwDDYhixucUi6vEY
EH5JtOwVVQ7bubT8OFeOoJwVxlzYz4UoqxjKDWymarTzu03AUIT0PXPece94bJAK
mSh68ItQe3H8tSPMubERWz2tEV3lVkChDGXcC7BYQmxHseolxz/qzCtJ0iX9BvZR
```

```
dniZNeNJ/Cu8M2pDp47zdNFXzf/Q/sQ9pQlws22G2g119rWDneBku9n1vTP80/er
SB+VLTBjDiArlCY5y9+BG8wbscExJySoQxkB9j/nlMzPY5rgk0SyxsNj9GbqH+hr
EjS3/uacNwSLxGcOT2E9Teot5pfTEO6fQVq+35QhfAlP8c8jze01W/+u+wXu1Ui9
azRSzYtCHanGyyet6U1mlBpAkqkZzH6t3CA5czc9i6FbzjvFVZnbRUZIRzfISYew
lF5WqgTn2iYVdxagPRvLF5kjd696brGW9d5HwirCVGaK04VsXWlAblB9wsF9BBMB
CgAnBQJXdYAFAhsDBQkHhh+ABQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEKwQ
ezhmktrdWigP/3QWl7aO81BUWyby4HEhN4SdAoWGY/FLqO4mCtuplcnMgRUCSiL9
l2BSCTMCtUcdSWtYwOgSChN2mMsdi1U2FNR5HvNunYR/pFdqjfQurf1ZmKVeG5/4
uuKaOxMw9e8pK5uYAfs+O7gr8gu/f6/Drp7NZk3/yVKpf4WCY9oX9TA1q9O/11nN
cwS45U/d7YP+N1YM9cBXa1DnDcdfm0BlykzouAF0qd1Lwi/tmLENvybD3+2c2WsE
rlFZGSa5ZafO0tTIWXh5k6wh5FdRRycrnSyRK3B9N9+yaXfMQ0XpOypa8dqQEnCi
IsngDCJPxtTrhMWKhBFRUMzK/WZTDboTQSQDK+YVRrE4K8MtoZSKwZLV2r9O3TpX
kpbKsPVYmexerfdMeZfjZMF1bC7BmEs7jciH6JjbqAoAPnHzN0481aeNarINSViX
PQWr2mp9qShei2/RavLtx2ZNrvmGW72ZKpF8E3WWUDpBJqFVeGNRvOm3aZj8o/Hl
ewtNjcT4ouJfqlfKiULv+g7ANEMDLQTFDTg5twRdvmZlB7oTBsavf+LwxPIXhH32
IR7TX7VeicMMxmZnmZK2ANT/QBi3laf+ojVHvB+f6D74eLNq0Zqjfi/3UFNYsYjg
E+YgCqEUBpHbl61nOHwGOSsQwfap2uKKlzukD/KxH5SPBC3DYGBI+KCbzsFNBFd1
gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+hV6XulGA
HAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go7xHIxgFj
C046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FKVYR/j9ue
nEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQlQ1Kou+3
dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjypUwgp0MT
o25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGwsMDyHNqyJ
eYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6QxaZje9YSZU
ijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KDOSn5CbmX
pAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVgroUVtprs
mHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs/Hd981Fd
VghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQABwsF8BBgB
CgAmAhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZm4FCRNDGegACgkQrBB7
OGaS2t3y5g/7BFXp/fdanzuQPToJTPen7AVwhLloKaiYhG3GjdXfMPLvu6UtaaGm
qynLolUNNooobptFqc1G9BKoAghQrta7CsDHtsQF2xyc3Mfu0gmpL/7X5a7sFIeJ
j08UjfweHx4DSG4LEZgNaAoWFjZltp4+8cqijkAHXt+r+1ayQG4VVHOWyXXqmSH4
9HqtbPcPyRzxdoVLeshZC9jmhHhhKqw/LwGyipWSOUKQDjWarBwdyhNmWCaLvxH1
ndMp4tq8DPGC3G4T9tYAbANrn7nKfZgHebMSzMw9kSp0L6QvwwTDjJyIWz85WyeH
WHeBysDaBOit3XDlehUew27y7N6a9hQSYjnXuwvre5mjDIOqJon/31R6ui2Z1y9P
a+bC11hbLXXh9tLCXRuoOt6thh9Cq5X1a76PPpEv30o3bpsb6l2hbrut1OKezwvK
l7txito/jfMiWfsZHA9O4SoM+8GnmVingHtZ805n1T4RddJvT/vaqplfI6zf7jmf
a69lALP420riFOQcwntNUM5tVmFUZsnFp2YRd4Ls7MiXVjtABahlSbb94l5WSVc0
jrOLDf94edvzk4R8i2Ob8CfVZNqEsTR6bHz8dT7Q+xQzEdjUujyyZY1UUl157Qeb
OsHjhCtuZYCI04X9hZ37nKnZXSxRlRDCnt5BEiyFu2WD1RscUe6PcVDCwXwEGAEK
ACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCaNdlPQUJE0MYuAAKCRCsEHs4
ZpLa3XCpD/42DrcveE+q2ulrAIYPDlUlHiwIMejqBDRm6zmr1KSAeb4E6/MFcP4s
rXSSscMlrqG6NVynjNCXjD2YzWii68EwoXLJkgoD3r2ifzkV62EX2MIEeNZAVwuy
KNxorzmy6bhuWltRYNK/hITs2AG5orOk9ADEJ8PixKymrWlhesPaWX6Yhp9/tWaC
```

RHOSRiLbRVaJ+7sqT88urLmkV9Hqx949Zxv4+cgBVUGL6WXKsfWhHjbDMNJnozWB
SZaIJznLApOM8z+1DNrqUYyfR8SkF4IOVmg6HDzoyuseJJ8JvMAlkvT6F9VBq/iE
yeDYdEEQxwHwozKrEx5Ybxl5mntbqwCXy6kHSx2+/3RZWpZQ8K29YP9QEk0KeGF8
9Vap3jjNrX4u3cuRNQpeblQc4uFn3Nzaj+cVV4YzcRw94NifecXpujSvk8XU2ytJ
/JgMBxPIBKglN4eEMet9b4FRB5XeBdPAm19/LXyb4IlIipGNXlgNz/HCuBzidzHT
QQdqfA9rZVx1hwFr7AJCVqWaXVsx1oEAhKqpTtsLMyj594DvnRuwKw5Vse+1eydW
MIHYdbxmJccsTGIt/hsOpc8zfm+QYk5752jshhOKEBy+Ey3QZI1WbO547NOb2Hwr
Pgt7fw2NCKMPElSu98zmneFPhqNHf7L5urBe5gADj81E8lm6t/oVxcLBfAQYAQoA
JgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJnABcLBQkRa8qFAAoJEKwQezhm
ktrde3MP/13CLWp99XvRROrzD/bWOfWjAenT2PE/tYdOY9YcTQFbnIUhaVUDWAo3
pibR3D4u9LlY4olpGfJ7BTIHFa9myfpaVvmrNjueYI4omli24JQ/CKqNdY8Qzxz+
/QyiNK7Aw5cEBWIu84WGB1SsefWWT3rZe9YBb77gNcWHZ15pXTXrcgUxGY4808MC
I9YFWq8EA0iHawtFnmB3UFfClWt37Hy3PKvr1is3uG60+ULI8RQz3/+ZwSG8U+xt
b+I7H9+gITc1eFCb+tIwp5xWflyxcFXYk6UzOL7y3Fg2tIEuSNtIHUC9NDVobf6c
I0KAzZcMvKiPQiuBnVOjgDLmCZM5H6axj9x+gi4oVh6ea3HLqMzyjm5JkeCGgKWv
H0gD3yGEZDvcbavkQOle5T+4JefndKzCPrluX0iyx+oQiiOL8WieSSkSB6BsZcUN
SeuGJwM79Y7Oqld/YVrQNBZj5Vz+m3nZ+0EWDDMI0hRgMpSEIc+dnTC0u103Z+Rc
c2IJq8INmU653sUcfCZE12ParW4rF7ib6kViYrABT8f4e2TP0aOyP5kp51ied9qL
azaBA6tt/C9X1V2EJZK4srXtmcZO2Im45RAiVXyfpBAmmiF3eZWCbKe7qBC4rDRh
LZG4RQW/S86Da0BID7gQz9IFSkaG504MsDhvnA7iAqaHUHUepCsiwsF8BBgBCgAm
AhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiQFCQ+P/Z8ACgkQrBB7OGaS
2t3AwA/9GkXKUgvjKGCxwE4SdDt7c2jw6to2TTP9iFJ3Xbk3+5BURT3gkZCuu9D7
gt+97aVo/B4EM7Xz8DQKyY7Ic9VAwDRra/Hwi1V0hw1zyIWQ/gAnX3baU6qLRWHR
vVR5meV8r35C+rg9DaWFYmvS7PIv9LfxESwBPUjbmx8k4/5EJpHUwf12bzkTnot5
7q5lHxKQa6IvqQak+Hp9ZM2KPdsgKO2HWJJIIvYcI5byW9zBKV0O7YR8gtRAJKp9
IbtsXx0WT6cqHOFVc5SSzdcaMt0gLFl7BTnJyvKK2l9GABGBmzYDjeCyF2J+Ippf
oqxqfTe6EoOsuEMc2PbLTs9SsWjyCC2VGlX8+uUH9SoKwL0VQ6LFsP6fhkVKqi/a
rB6UuPR/iZnrKIuxMNQ4U+t2Q6UdMlmXsAXTNdkwzoK9oJRokIrH0ZV1KtH4sjjA
tCic+tOddq+GQLiKe2WpJfxlA0uESCB0TxjAwQmfn1H+dUhPeLlbNimHlH0/hXPd
ifuNGozzADIRseQDyzjl8xGL1qRZLD3cfmda6RyZ+S3dQRuaRrcFCDccpY/pO+F8
jbx64zyqqNs+KV+SkQGOcKFhWTZGCfQ/zMDtDmQKjb3eTAkv1zdEOMw9zEjjmS0q
8FNl+2wO3VnvXwvBbtDdVCIaIq+jVcsy5XtnnV+bJ19Q9yue/XvCwWUEGAEKAA8C
GwwFAmEyoZoFCQueVRUACgkQrBB7OGaS2t1uHBAAhOYVvrtchRmzCvdNER1DtkIs
bgQPJ9OxbyfvmvoD06qxH7PrycLZKbt7yYpAUU/CMc86GwaEe0I5Nm1CTs6NvDIv
g3e7EPIS859tyQflbM56NlwbsopCuoCJYknuroIf/M6dW6vJKNXLMmnL/AtalUBw
X+5pblmGUUJep49oTOxQEnvnuqyvaGjXgFXix5PVFJD2ed5NnQeFpvfCpc/ioNOj
z7ORO82j1ht5nWqPraXX5AYhQFM/kwR1cK4LV7gVDd/q+dfGYHzpxQ/HtyX/Lasi
N6I52QqA95SM1ZZLPFLaNh6EvnB7uC9pLCYS8nvilX7/cez5PFff1e1gXCOT0jv3
mJ2exLmXV0BbfKgjccFCxhrdRLtukfiDfJkySy1zdscnpfng8wJ3xKRv43cUTz7M
Z24OYNMqK26aJZVXEQUYjCwsBylY/F5wjYAwgwZ8yF5RFix28P/K8JsIHb3QrAJK
sNWQAb03ZWis3N3spR5M9Mw3VuDZ3WUXq7mxB5M3kpVoZ3vETU5cwTbADYNPf4Sw
BDK2uIVtxabezxSBtz0FcyYoF+OW8q7r4WvoyC9/+3GfnozZLJcEIVDk4W2pMW4A
UhG/6drKTm3HkSDWIDu7d1sHWMffLEYfUHtN5DKkDkGoPfHvZvu9teR5yLfUrPTf
ktihPn/JMrmwa9pwi8LCwWUEGAEKAA8CGwwFAmCavPcFCQsGcHIACgkQrBB7OGaS

```
2t0uaA//UWRaRiHEAKeRqBG/T2ak+XZJNu7QHfNgoUEAub9Zru8oPPXx2AJLcHEN
KWmeFlLxADdWOZs4Bm9oOew3VQnR/dBqjnXfob9Rc+eYUjA3rXazM/QrqcU8Syi3
MjNGUmjdL5aQF+IppAMgOBLG1TEnM7C5/PvrGJuYpGEnkKEwMK/GYhqg2V60pHEV
Pvs66mefJpCzbZSy56qtknSt6yBNWc14XgDX6VTn2kW4CV/3vVJUuvjvYs9SPyY8
mKEXa6QvUd3PcXv6RiWk4lGYuT1+jh2VkcFQ+JnUwv9TbKFB9b5jq1bvW9+LMDEl
YXux7pBP5RPk+OLpyiExIRFWhi3x7aMWOzQ+I9yuNTeYkTHiEAQRUhs/1Fh4oLgI
v9QZgC0mRSN3zm8plQdivs1ZlAosAqqkA9BQwqsgosQe7P92irYIJqay0si9wGCD
wSMsmeXdIF6wW3/UMJZl66aarPeiZApGX0QdTZwjMh/QK/8gTKyeZulKmNkNfwWq
O17OirWqLKssVHtg3VUM8EIdh+oNqDDXSeWtYUmpPpWp+yWZ0x1MFFZhUQHQZTGu
TIj4A92LQzbrfj/jXRvWm2SrJMivUoiDUn+qxKIpVwFlI5gVb+uyTFhw89PCkphr
JwRi052RLoU9yd6Ek46UH4XfZZWrZuzY+zzB7oqGONphLgi/h3DCwWUEGAEKAA8C
GwwFAl771b8FCQlniTUACgkQrBB7OGaS2t2/MxAAjoEGPdzavhsOlXdPCRd1D5QJ
r8T/NSEV2z1cp8ZvdrkjNF09TBP4qsBnKJiuvY1Iw7OGX9W2okvXxgJizE45v9MH
WEMz4hmIjmAfRwcqENgpOc1IY/T0/+kkCW8dB6d30J1kT0n2PCRzN9L5vPqZXGTG
mLvd9MOjH1256w4uxLb+e1HMDTCqEN1ppq9G+EAR/29q8JZWs1marbZZWxSWcg/E
1YYbNafzklgjq4CLh/j8AEWSvLr39zRy9uvQ/yqAKZ4K4aZfh/SPupGDvsD6ZK54
EPHxErQ7aiXTbUHtvwhxWLOP6WmxFA3Shr6L6YUb6jq+0PVliFC517g3mxFHJtwt
yXGNIKhmzmr0l9OlsHafulJ/9QPfK3Ce32SkPhW/11MYA8HzduMv5Arp7cBczXSP
EUTmNIVKv3gTjSQrzRhwhHmMuqyDZ/rXQQ1jl2sxIDjO4MUMvVjYKF+OCNm42gVs
8ca3/wN9ZNU6hyFWeKQDuCAqPPbT5GO/DKseFEwB+07wwyH1RXbyl0v4fneg605X
S7lqhNtw2p1hDL0HYHDiV+aPZ+LoOmX6+dmnqE6bQJaIlVb922KWmliO7F3DkqP7
0jFlhoE1gfiXWkxP4Gy8wOobNfEMgvz02djkGQy+oQqeNdIcZFZgzPTGKB/nVgpt
9CcRDWjPltFCd2e1FBbCwWUEGAEKAA8FAld1gAUCGwwFCQeGH4AACgkQrBB7OGaS
2t1PIQ//Qc5VYfBCxpaMysaPQ44wXPEZSjxIGZhhMGzb1UzzAEYOw+RgKN5nNTXq
L2KoOkOrGnKqZOKByMdXwIPH/rGwwEsbbIpopnibf5ic5B/+xCTIK+qLIwX2ZLuk
NhbL6Y+E+7DxMMh+KqBWHONKkgwVY+rFWOfoops839ABKvc9/Ry4/qqkcb40AzpD
l1iQJ5vK/DMuaDWxWeKXqJLIl3WMGPcPfheuBZL1u7LEEHYKMgzvpbF81WIn3MBo
8jvxf2/o+kMafSSDqgvOu6yu8GOhmScpCbRJn7jV/HrG+tM+zy48TN6/MkGWSR7q
TD34pqBjyatVfVl6dGD6xj/i/Emt5hZB6qXruCDH7AWMoNx+FkDubs4sc4PKysZU
Itya6KdQFo2UeYsNwZhdn6QwKhd85um4JUHJCY0mARvjsQgWXH/5MR40cow77bbE
vVq0XNd+QRVlyT42CEtnIUOFLeDVuZrum5Tuvvna6ImMDoi/z6QcNeL79XsY2m6I
QVRiHr1BDb/8JLkfnWiwL8GRv169Kf8unx0y5u1YBpcMYkyDD2+pnnk3TY0rR+8X
8goecaS8fbyu/Q48K85ZMD8wKW/bzLQ+tK9y8xed24u2QERftMhIw9b6f45Nrrf/
PhgV8RnuwUusSbdDe8kw3eYTmLdzD4kZc9K7SdO2CqT+hm//9JI=
=uGHC
-----END PGP PUBLIC KEY BLOCK-----
```

Tasti precedenti

> **Important**
>
> Le nuove chiavi vengono create prima della scadenza di quelle precedenti. Di conseguenza, in un dato momento può essere valida più di una chiave. Le chiavi vengono utilizzate per firmare gli artefatti a partire dal giorno in cui vengono create, quindi utilizzate la chiave emessa più di recente quando le validità delle chiavi si sovrappongono.

Data di scadenza: 2025-10-04

ID chiave	0x AC1 07B386692DADD
Tipo	RSA
Size	4096/4096
Creato	30-06-2016
Data di scadenza	2025-10-04
ID utente	AWS SDKs e strumenti < @amazon .com> aws-dr-tools
Impronta digitale chiave	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 692 PAPÀ AC1

Per copiare la seguente chiave pubblica OpenPGP per l'SDK for Java negli appunti, seleziona l'icona «Copia» nell'angolo in alto a destra.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Hostname:
Version: Hockeypuck 2.2

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

```
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJnABcIBQkRa8qDAAoJEKwQezhmktrdl1MQAIwEuDar3OTxkfTa
cPNKDnNzxaWqrxZ3FTQ+PyrHhQ6usxxrvDKJS+uCjE9bmWHVFU1R4yQNF+72lJdw
5UhX0u+ZgT9afApE65uAZuwLhPsz8upXT8C6VeKXh3shdw7qXi2hrwtM1a0Pls4O
Cs2C9rLUDMJTySrVDDVwpnaAB+8DcFrs9bIt5Q3gdOUatdzDvcb7QKh9jUvzCpbE
cInb1epDN5MRzowMR4iU2VV1RzLxCvm7CQSyXfgf0DFLkXWiknhOq9eINmytJFG/
ntFdiZFkNZ5hP709loywdfNrmqB6PsF8BPGFh8gKw1pjowrfHpv6cNIqShmA76LT
3OHViOlqGFB7obffq//eZGPR0oYJFDrOdD2CFRoHnP3N++AfkA4SRN7eXwyoz6Pk
Do9WNIEEkAcp6PGvv7AokogDo/4OqmxgC6fN+3BTOstWpv4FlD4NxOZWsTs49wxg
kPlCCVf8t75aZZkcjXng1eClZZQ5SBlRtSB7gMqtP7MIn2J5w8spNbs5xQvJc76u
NvzwEasPkY+UcHd05RddOUwoKqDerLUG7YqdlNCJoQR1mBIgZButbQlMyaZcmQqO
iR0kwDi9h6Dl6fnUb2dFCNJw+eDHvsjG8HI3IVZMl0bUQ2kmmwr1O2YQ1ynJQm0l
lMllI4hFU8/lHNHm8ie5darpVXQEwsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiIFCQ+P/Z0ACgkQ
rBB7OGaS2t3HVg//S+/KbeOBf+nCdHsrWtp9kxvWIpAGvQhIbxx1tp/impfm+5Rm
fKPD0KX+g42fuMmOdDE4gjO4GjGd7ZY3bx+0zbDSdVebzmYCbPZ/BDP990oPKidd
w6Gl8PaIyqfuARKOESBETvAwNgwO4t2ocjs4pYZV+CuHvESYpqkuHjmHtye6ajZW
Mv24NhjVo4EfP33dPugTjXLjeuGT7qQpsYV3a66juHmPVkXwuPqxh9wTNc5TU6FG
UPSfIGMPL0xha7Rg2i5zvRaAxx4bHqGO8IAz/l/E/tJkV5xnt494HQam9UDbiFI0
Q0TSve1R6S45/UjQW6cycyduHtk72s9ipa9YM0ilTdLgKMWFjzYv4h4qeYvLw3oB
JGQew+I0I4dIrwL/TKet33EuFfwmyT9MaJBhqV6geFaQ0uVmwvzpAcvxIoSqSpkJ
B/kASqCEM/o0QZLuWc56cDsmMisD0ouVPt+c1Zk7AWLlf6j8LKYTbK0QxLRh/eeZ
jhSf3HnpaCfonbOoHmeo5d/o3EZ0AiA4GbT3xgScoIgXOT7KGgOWmtWkdyd3Zvl/
o6q6Hwpg4RsqCkwnfNm5ZIvmTAYXWc2hiICSicxrP0fek5Cc4xVgJR5RMNGyI7+m
ut1SE2WvLhMCwEyl5ecWF0tUze8VBlWkHJp0Y4k2ado39Zq/DZrTRQYEVrjCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmEyoZoFCQueVRUACgkQ
rBB7OGaS2t3axA//dgcz5T4Fs7LWdIQB/KRnvX64IzaGQcwt3FBhYH+sFaSaD+lu
752vi3jlGxMBKs2NFxk6e2U8xBEir3vfKYd+mZ5L6egXC9MYfvMO/UFEFH3A+t3V
dTOJK4RQcaL9+rFRVdDmZuifN9OFfy25d66JCZ5OiqgHTQViVmbrGw2cQdyNWXrq
YLgg+gktadkmzis4J6hF/le8NOBfrG3n+QthFl/v2ppYYW9pmmxzUIf6tAlR1Vr8
PhPukjuFfrPLRL3XPiK4LkdlSI5MX7Llq4RkcZN1NY1LWS8699wJOLRcr8aQYvzZ
Jm7tfZUaekJ5ScXJWJEaWT4poMbWxXINj6VwE+DqKWvjkzoxBXSIdLk4XThA1dIq
3n5SfMkfuWV2A2xHqJtEzI1XeTm/d/JRxIG8hjIs5FNMGJUSNANJuTVA2putCVf0
JbP4QlafXoEVOYwW/EFJY+brjQadwc/knf/QxszDcKb3THuTxR80A0h6ZysmtLEg
PCaD1xUCDdr4P7DGOtV7yMaDR6O8QKmb7TKzCKCPnHouqPT0DhSB2MRq437+mfSe
EGga/sPMxe0z8ugO2CnrCf3eplU3ZOy4eeQSThTKe0Hp5YlsF1cdZSvjt7GJaHR2
```

```
9A22nAJY9PojtlM+0Dkr/PH6r2brv3sEuACRNhzqCWhAve+zVnVLeb+Fk1rCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmCavPYFCQsGcHEACgkQ
rBB7OGaS2t3fqg//asHYSTBI4IoLibSF5ZJ8NaLo4EqgRtsO0W1zr8fCoFbxI+yI
qWrNXR7eoFj8tWO7Tj6S0kQr4QZcucLeILfox6CtDzO3f3WQTH9m/0si5U4Jf3RA
gBd0vwxVSSSNEsIUUfylOBHLVwlhaTVfh1h1dZhzb18LA1Vqu0100GGxvaG3dUl4
oMSRSK8EeaSO4hIM8ScjVyhjsuPRm1jlwd1EXZ5mkHRGRMGmwi3XysJjIeQRFmuG
a9uPes7I/K85r7X91BLz+G/mkSZsrHxzLxF0mSZtQdhq08GyMeQlJavs7sb1UVbg
ag30JEAjgqRsNIQOMIvl2/amrRJ7DUyQu5YkZQsAyCIhSVZw6zlJlkYVKSw1Cu1I
VX1n/Aahx5wwaQZA1dDTolX8WvL9O/KmEGWbKUSov40uoRwSOeXP3QhW75kD4zT0
n3pSUc7tXka8GAz5Ar+r90l4wc9as2WjWADo3CX8cF0zQOrN1naMQ++xBIAG9ms8
y3L4ECoRqvjCpjUAfhflxwSM7uclCgFBINFKSLV+QGxzWfHjg3+bSQd0hrRn9j4z
Di9aJmFESQ6iykbUjiUFrcI1NUCFKz2awaxh+Sq8UkYcvuxljxdK/zYYEG8DSdQ2
uwlssHCjYVpAb16hq1EAaSqnhv86020G4Z6JCg3AUZt9RlMBpK2Pn/NmPSzCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAl771vAFCQlnimUACgkQ
rBB7OGaS2t35Jw/9GhpQquJVUCAsc4az7+ad8q2d9OUKXOLl2x0j3/ofS8umZJYr
8KXZxPqBqvLjlUyAB5Ur4fWVBdp9iP4Vj74jmRih7YatK8heGmNoUAnI1/90qV5O
ypFObfvWLxvcUNizmS/LYKdNuYcHwyw4bFyTz9gd3XH6Dxak7YiAXz3JgJIXcIB3
ELfpsduzpncLeVwz/dKhYX5iJ7Y/xd4Ew8Ian8FyNDNRwcXobTpPAMNiGLG5xSLq
8RgQfkm07BVVDtu5tPw4V46gnBXGXNjPhSirGMoNbKZqtPO7TcBQhPQ9c95x73hs
kAqZKHrwi1B7cfy1I8y5sRFbPa0RXeVWQKEMZdwLsFhCbEex6iXHP+rMK0nSJf0G
9lF2eJk140n8K7wzakOnjvN00VfN/9D+xD2lCXCzbYbaDXX2tYOd4GS72fW9M/zT
96tIH5UtMGMOICuUJjnL34EbzBuxwTEnJKhDGQH2P+eUzM9jmuCTRLLGw2P0Zuof
6GsSNLf+5pw3BIvEZw5PYT1N6i3ml9MQ7p9BWr7flCF8CU/xzyPN7TVb/677Be7V
SL1rNFLNi0IdqcbLJ63pTWaUGkCSqRnMfq3cIDlzNz2LoVvOO8VVmdtFRkhHN8hJ
h7CUXlaqBOfaJhV3FqA9G8sXmSN3r3pusZb13kfCKsJUZorThWxdaUBuUH3CwX0E
EwEKACcFAld1gAUCGwMFCQeGH4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ
rBB7OGaS2t1aKA//dBaXto7zUFRbJvLgcSE3hJ0ChYZj8Uuo7iYK26mVycyBFQJK
Iv2XYFIJMwK1Rx1Ja1jA6BIKE3aYyx2LVTYU1Hke826dhH+kV2qN9C6t/VmYpV4b
n/i64po7EzD17ykrm5gB+z47uCvyC79/r8Ouns1mTf/JUql/hYJj2hf1MDWr07/X
Wc1zBLjlT93tg/43Vgz1wFdrUOcNx1+bQGXKTOi4AXSp3UvCL+2YsQ2/JsPf7ZzZ
awSuUVkZJrllp87S1MhZeHmTrCHkV1FHJyudLJErcH0337Jpd8xDRek7Klrx2pAS
cKIiyeAMIk/G1OuExYqEEVFQzMr9ZlMNuhNBJAMr5hVGsTgrwy2hlIrBktXav07d
OleSlsqw9ViZ7F6t90x5l+NkwXVsLsGYSzuNyIfomNuoCgA+cfM3TjzVp41qsg1J
WJc9Bavaan2pKF6Lb9Fq8u3HZk2u+YZbvZkqkXwTdZZQOkEmoVV4Y1G86bdpmPyj
8eV7C02NxPii4l+qV8qJQu/6DsA0QwMtBMUNODm3BF2+ZmUHuhMGxq9/4vDE8heE
ffYhHtNftV6JwwzGZmeZkrYA1P9AGLeVp/6iNUe8H5/oPvh4s2rRmqN+L/dQU1ix
iOAT5iAKoRQGkduXrWc4fAY5KxDB9qna4oqXO6QP8rEflI8ELcNgYEj4oJvOwU0E
V3WABQEQALzM0Cs9Zvd08xOEvbEBj59LrS9d0HVKQ61gmkNakWC+jR35VD6FXpe6
UYAcBLrEbVYfKw9P0p6MhFKAsb570JoznKGzE1rVYUZQzhD0RKje35rvkajvEcjG
AWMLTjr87pWHeD0389ER64bzORncfa/l+YP56PI+CThb2wUvTTONGJkPQUpVhH+P
256cQL/Y0Fwu4XLerpwN+YKgMQ47raRcydobPeSfMQr9fVKRyOzFEOrvNpCVDUqi
77d0gLDLjHlIlDyOX5554S8XYLb91eYOiFvnu2pTCKiiExRCSYK29mAQePKlTCCn
QxOjbmBbGS8mVIkpQ5vpvXvzpY3JIjMXaDGqWSQSYGXhECyxCR5eOtKYbCwwPIc2
rIl5gW6yXyw9pKmj5XafTP7YHTvRSr7CZ/VLkDkWl6AfQ9nPOg1mjwjpDFpmN71h
```

```
JlSKMaZkh0QGV5FW3dK+GLwxiWdqx3htbZErWyvumWQF/xBF7puKJBEXcoM5KfkJ
uZekBwcnVkfNFF2RdkM1ALq8InGzLXc7ROuEm0BXVirfju7JRtWLb3UhJWCuhRW2
muyYegSTkag5MduD1IJK37GL8WIlAL65taYgZegUoxHdSaEOefOhspxuduz8d33z
UV1WCFhi+r/+BMCQmTRbF8ao7fTC1dGd084DRP6qE/dMT4u0ZEn7ABEBAAHCwXwE
GAEKACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCZwAXCwUJEWvKhQAKCRCs
EHs4ZpLa3XtzD/9dwi1qffV70UTq8w/21jn1owHp09jxP7WHTmPWHE0BW5yFIWlV
A1gKN6Ym0dw+LvS5WOKJaRnyewUyBxWvZsn6Wlb5qzY7nmCOKJpYtuCUPwiqjXWP
EM8c/v0MojSuwMOXBAViLvOFhgdUrHn1lk962XvWAW++4DXFh2deaV0163IFMRmO
PNPDAiPWBVqvBANIh2sLRZ5gd1BXwpVrd+x8tzyr69YrN7hutPlCyPEUM9//mcEh
vFPsbW/iOx/foCE3NXhQm/rSMKecVn5csXBV2JOlMzi+8txYNrSBLkjbSB1AvTQ1
aG3+nCNCgM2XDLyoj0IrgZ1To4Ay5gmTOR+msY/cfoIuKFYenmtxy6jM8o5uSZHg
hoClrx9IA98hhGQ73G2r5EDpXuU/uCXn53Sswj65bl9IssfqEIoji/FonkkpEgeg
bGXFDUnrhicDO/WOzqpXf2Fa0DQWY+Vc/pt52ftBFgwzCNIUYDKUhCHPnZ0wtLtd
N2fkXHNiCavCDZlOud7FHHwmRNdj2q1uKxe4m+pFYmKwAU/H+Htkz9Gjsj+ZKedY
nnfai2s2gQOrbfwvV9VdhCWSuLK17ZnGTtiJuOUQIlV8n6QQJpohd3mVgmynu6gQ
uKw0YS2RuEUFv0vOg2tASA+4EM/SBUpGhudODLA4b5wO4gKmh1B1HqQrIsLBfAQY
AQoAJgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEokBQkPj/2fAAoJEKwQ
ezhmktrdwMAP/RpFylIL4yhgscBOEnQ7e3No8OraNk0z/YhSd125N/uQVEU94JGQ
rrvQ+4Lfve2laPweBDO18/A0CsmOyHPVQMA0a2vx8ItVdIcNc8iFkP4AJ1922lOq
i0Vh0b1UeZnlfK9+Qvq4PQ2lhWJr0uzyL/S38REsAT1I25sfJOP+RCaR1MH9dm85
E56Lee6uZR8SkGuiL6kGpPh6fWTNij3bICjth1iSSCL2HCOW8lvcwSldDu2EfILU
QCSqfSG7bF8dFk+nKhzhVXOUks3XGjLdICxZewU5ycryitpfRgARgZs2A43gshdi
fiKaX6Ksan03uhKDrLhDHNj2y07PUrFo8ggtlRpV/PrlB/UqCsC9FUOixbD+n4ZF
Sqov2qwelLj0f4mZ6yiLsTDUOFPrdkOlHTJZl7AF0zXZMM6CvaCUaJCKx9GVdSrR
+LI4wLQonPrTnXavhkC4intlqSX8ZQNLhEggdE8YwMEJn59R/nVIT3i5WzYph5R9
P4Vz3Yn7jRqM8wAyEbHkA8s45fMRi9akWSw93H5nWukcmfkt3UEbmka3BQg3HKWP
6TvhfI28euM8qqjbPilfkpEBjnChYVk2Rgn0P8zA7Q5kCo293kwJL9c3RDjMPcxI
45ktKvBTZftsDt1Z718LwW7Q3VQiGiKvo1XLMuV7Z51fmydfUPcrnv17wsFlBBgB
CgAPAhsMBQJhMqGaBQkLnlUVAAoJEKwQezhmktrdbhwQAITmFb67XIUZswr3TREd
Q7ZCLG4EDyfTsW8n75r6A9OqsR+z68nC2Sm7e8mKQFFPwjHPOhsGhHtCOTZtQk7O
jbwyL4N3uxDyEvOfbckH5WzOejZcG7KKQrqAiWJJ7q6CH/zOnVurySjVyzJpy/wL
WpVAcF/uaW5ZhlFCXqePaEzsUBJ757qsr2ho14BV4seT1RSQ9nneTZ0Hhab3wqXP
4qDTo8+zkTvNo9YbeZ1qj62l1+QGIUBTP5MEdXCuC1e4FQ3f6vnXxmB86cUPx7cl
/y2rIjeiOdkKgPeUjNWWSzxS2jYehL5we7gvaSwmEvJ74pV+/3Hs+TxX39XtYFwj
k9I795idnsS5l1dAW3yoI3HBQsYa3US7bpH4g3yZMkstc3bHJ6X54PMCd8Skb+N3
FE8+zGduDmDTKitumiWVVxEFGIwsLAcpWPxecI2AMIMGfMheURYsdvD/yvCbCB29
0KwCSrDVkAG9N2VorNzd7KUeTPTMN1bg2d1lF6u5sQeTN5KVaGd7xE1OXME2wA2D
T3+EsAQytriFbcWm3s8Ugbc9BXMmKBfjlvKu6+Fr6Mgvf/txn56M2SyXBCFQ5OFt
qTFuAFIRv+nayk5tx5Eg1iA7u3dbB1jH3yxGH1B7TeQypA5BqD3x72b7vbXkeci3
1Kz035LYoT5/yTK5sGvacIvCwsFlBBgBCgAPAhsMBQJgmrz3BQkLBnByAAoJEKwQ
ezhmktrdLmgP/1FkWkYhxACnkagRv09mpPl2STbu0B3zYKFBALm/Wa7vKDz18dgC
S3BxDSlpnhZS8QA3VjmbOAZvaDnsN1UJ0f3Qao5136G/UXPnmFIwN612szP0K6nF
PEsotzIzRlJo3S+WkBfiKaQDIDgSxtUxJzOwufz76xibmKRhJ5ChMDCvxmIaoNle
tKRxFT77OupnnyaQs22UsueqrZJ0resgTVnNeF4A1+lU59pFuAlf971SVLr472LP
```

```
Uj8mPJihF2ukL1Hdz3F7+kYlpOJRmLk9fo4dlZHBUPiZ1ML/U2yhQfW+Y6tW71vf
izAxJWF7se6QT+UT5Pji6cohMSERVoYt8e2jFjs0PiPcrjU3mJEx4hAEEVIbP9RY
eKC4CL/UGYAtJkUjd85vKZUHYr7NWZQKLAKqpAPQUMKrIKLEHuz/doq2CCamstLI
vcBgg8EjLJnl3SBesFt/1DCWZeummqz3omQKRl9EHU2cIzIf0Cv/IEysnmbpSpjZ
DX8Fqjtezoq1qiyrLFR7YN1VDPBCHYfqDagw10nlrWFJqT6VqfslmdMdTBRWYVEB
0GUxrkyI+APdi0M2634/410b1ptkqyTIr1KIg1J/qsSiKVcBZSOYFW/rskxYcPPT
wpKYaycEYtOdkS6FPcnehJOOlB+F32WVq2bs2Ps8we6KhjjaYS4Iv4dwwsFlBBgB
CgAPAhsMBQJe+9W/BQkJZ4k1AAoJEKwQezhmktrdvzMQAI6BBj3c2r4bDpV3TwkX
dQ+UCa/E/zUhFds9XKfGb3a5IzRdPUwT+KrAZyiYrr2NSMOzhl/VtqJL18YCYsxO
Ob/TB1hDM+IZiI5gH0cHKhDYKTnNSGP09P/pJAlvHQend9CdZE9J9jwkczfS+bz6
mVxkxpi73fTDox9duesOLsS2/ntRzA0wqhDdaaavRvhAEf9vavCWVrNZmq22WVsU
lnIPxNWGGzWn85JYI6uAi4f4/ABFkry69/c0cvbr0P8qgCmeCuGmX4f0j7qRg77A
+mSueBDx8RK0O2ol021B7b8IcVizj+lpsRQN0oa+i+mFG+o6vtD1ZYhQude4N5sR
RybcLclxjSCoZs5q9JfTpbB2n7pSf/UD3ytwnt9kpD4Vv9dTGAPB83bjL+QK6e3A
XM10jxFE5jSFSr94E40kK80YcIR5jLqsg2f610ENY5drMSA4zuDFDL1Y2ChfjgjZ
uNoFbPHGt/8DfWTVOochVnikA7ggKjz20+RjvwyrHhRMAftO8MMh9UV28pdL+H53
oOtOV0u5aoTbcNqdYQy9B2Bw4lfmj2fi6Dpl+vnZp6hOm0CWiJVW/dtilppYjuxd
w5Kj+9IxZYaBNYH4l1pMT+BsvMDqGzXxDIL89NnY5BkMvqEKnjXSHGRWYMz0xigf
51YKbfQnEQ1oz5bRQndntRQWwsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQ
ezhmktrdTyEP/0HOVWHwQsaWjMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCje
ZzU16i9iqDpDqxpyqmTigcjHV8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF
9mS7pDYWy+mPhPuw8TDIfiqgVhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+
NAM6Q5dYkCebyvwzLmg1sVnil6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNVi
J9zAaPI78X9v6PpDGn0kg6oLzrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB
lkke6kw9+KagY8mrVX1ZenRg+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHOD
ysrGVCLcmuinUBaNlHmLDcGYXZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKM
O+22xL1atFzXfkEVZck+NghLZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7
GNpuiEFUYh69QQ2//CS5H51osC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02N
K0fvF/IKHnGkvH28rv0OPCvOWTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+O
Ta63/z4YFfEZ7sFLrEm3Q3vJMN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=bboB
-----END PGP PUBLIC KEY BLOCK-----
```

Data di scadenza: 2024-10-08

ID chiave	0x AC1 07B386692DADD
Tipo	RSA
Size	4096/4096
Creato	30-06-2016

Data di scadenza	2024-10-08
ID utente	AWS SDKs e strumenti < @amazon .com> aws-dr-tools
Impronta digitale chiave	FEB9 209F 2F2F 3F46 6484 1E55 0 7B38 692 PAPÀ AC1

Per copiare la seguente chiave pubblica OpenPGP per l'SDK for Java negli appunti, seleziona l'icona «Copia» nell'angolo in alto a destra.

```
-----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-----
```

Cronologia dei documenti

Questa pagina elenca le modifiche importanti apportate alla AWS SDK per Java Developer Guide nel corso della sua storia.

Questa guida è stata pubblicata il 1° ottobre 2025.

1 ottobre 2025

Aggiungi una nuova chiave PGP che scade il 27/09/2020.

5 ottobre 2024

Aggiorna le informazioni chiave correnti di OpenPGP.

4 settembre 2024

Aggiungi informazioni sugli endpoint AWS basati su account per DynamoDB. Per informazioni, consultare the section called “Usa AWS endpoint basati su account”.

21 maggio 2024, 2024

Rimuovi le istruzioni per impostare la proprietà `networkaddress.cache.ttl` di sicurezza utilizzando una proprietà di sistema della riga di comando java. Per informazioni, consultare Come impostare il TTL JVM.

12 gennaio 2024

Aggiungi un banner che annuncia la fine del supporto per la v1.x. AWS SDK per Java

6 dicembre 2023

- Fornisci la chiave OpenPGP corrente.

14 marzo 2023

- Guida aggiornata per l’allineamento alle best practice IAM. Per ulteriori informazioni, consulta Best practice per la sicurezza in IAM.

28 luglio 2022

- È stato aggiunto un avviso che EC2 -Classic andrà in pensione il 15 agosto 2022.

22 marzo 2018

- DynamoDB Ad esempio, è stata rimossa la gestione delle sessioni Tomcat, poiché tale strumento non è più supportato.

2 novembre 2017

- Sono stati aggiunti esempi di Amazon S3 crittografia per i client di crittografia, inclusi nuovi argomenti: utilizzo della crittografia Amazon S3 lato client e della crittografia lato Amazon S3client con chiavi gestite AWS KMS e Amazon S3 crittografia lato client con chiavi master del client.

14 aprile 2017

- Ha apportato una serie di aggiornamenti alla sezione Amazon S3 Esempi di utilizzo della AWS SDK per Java sezione, inclusi nuovi argomenti: Gestione delle autorizzazioni di Amazon S3 accesso per bucket e oggetti e Configurazione di un Amazon S3 bucket come sito Web.

4 aprile 2017

- Un nuovo argomento, Enabling Metrics for the, AWS SDK per Java descrive come generare metriche delle prestazioni delle applicazioni e dell'SDK per. AWS SDK per Java

3 aprile 2017

- Sono stati aggiunti nuovi CloudWatch esempi alla sezione CloudWatch Esempi di utilizzo della AWS SDK per Java sezione: Acquisizione di metriche da CloudWatch, Pubblicazione di dati metrici personalizzati, Utilizzo degli CloudWatch allarmi, Utilizzo delle azioni di allarme e Invio di eventi a CloudWatch CloudWatch

27 marzo 2017

- Sono stati aggiunti altri Amazon EC2 esempi alla sezione Amazon EC2 Esempi relativi all'utilizzo della AWS SDK per Java sezione: Gestione delle Amazon EC2 istanze, Utilizzo di indirizzi IP elastici in Amazon EC2, Utilizzo di regioni e zone di disponibilità, Utilizzo delle coppie di Amazon EC2 chiavi e Utilizzo dei gruppi di sicurezza in Amazon EC2.

21 marzo 2017

- È stato aggiunto un nuovo set di esempi IAM alla sezione Esempi IAM Utilizzo della AWS SDK per Java sezione: Gestione delle chiavi di accesso IAM, Gestione degli utenti IAM, Utilizzo degli alias degli account IAM, Utilizzo delle politiche IAM e Utilizzo dei certificati del server IAM

13 marzo 2017

- Sono stati aggiunti tre nuovi argomenti alla Amazon SQS sezione: abilitazione del polling lungo per le code di Amazon SQS messaggi, impostazione del timeout di visibilità e utilizzo delle code Dead Letter in Amazon SQS. Amazon SQS

26 gennaio 2017

- È stato aggiunto un nuovo Amazon S3 argomento, Utilizzo TransferManager per Amazon S3 le operazioni, e nuove best practice per AWS lo sviluppo con l' AWS SDK per Javaargomento nella sezione Utilizzo della AWS SDK per Java sezione.

16 gennaio 2017

- È stato aggiunto un nuovo Amazon S3 argomento, Gestione dell'accesso ai Amazon S3 bucket utilizzando le politiche dei bucket, e due nuovi Amazon SQS argomenti, Utilizzo delle code di Amazon SQS messaggi e Invio, ricezione ed eliminazione dei messaggi. Amazon SQS

16 dicembre 2016

- Sono stati aggiunti nuovi argomenti di esempio per DynamoDB: Utilizzo delle tabelle in DynamoDB e Utilizzo degli elementi in DynamoDB.

26 settembre 2016

- Gli argomenti della sezione Avanzate sono stati spostati in Uso di AWS SDK per Java, poiché sono davvero fondamentali per l'utilizzo dell'SDK.

25 agosto 2016

- Un nuovo argomento, Creazione di client di servizio, è stato aggiunto a Using the AWS SDK per Java, che dimostra come utilizzare i client builder per semplificare la creazione di client. Servizio AWS

  La sezione Esempi di AWS SDK per Java codice è stata aggiornata con nuovi esempi per S3 supportati da un repository GitHub contenente il codice di esempio completo.

2 maggio 2016

- Un nuovo argomento, Programmazione asincrona, è stato aggiunto alla AWS SDK per Java sezione Uso del file, che descrive come lavorare con i metodi client asincroni che restituiscono oggetti o che richiedono un. `Future` `AsyncHandler`

26 aprile 2016

- L'argomento Requisiti del certificato SSL è stato rimosso perché non è più pertinente. Il supporto per i certificati firmati SHA-1 era obsoleto nel 2015 e il sito che ospitava gli script di test è stato rimosso.

14 marzo 2016

- È stato aggiunto un nuovo argomento alla Amazon SWF sezione: Attività Lambda, che descrive come implementare un Amazon SWF flusso di lavoro che richiama Lambda le funzioni come attività in alternativa all'utilizzo delle attività tradizionali Amazon SWF .

4 marzo 2016

- La sezione Amazon SWF Esempi di utilizzo della AWS SDK per Java sezione è stata aggiornata con nuovi contenuti:
  - Amazon SWF Nozioni di base: fornisce informazioni di base su come includere SWF nei progetti.
  - Creazione di un' Amazon SWF applicazione semplice: un nuovo tutorial che fornisce step-by-step indicazioni per gli sviluppatori Java alle prime armi. Amazon SWF
  - Shutting Down Activity and Workflow Workers Gracefully - Descrive come chiudere Amazon SWF correttamente le classi di lavoro utilizzando le classi di concorrenza di Java.

23 febbraio 2016

- Il codice sorgente della AWS SDK per Java Developer Guide è stato spostato in aws-java-developer-guide.

28 dicembre 2015

- the section called “Imposta il TTL JVM per le ricerche dei nomi DNS”è stato spostato da Advanced a Using the AWS SDK per Java ed è stato riscritto per motivi di chiarezza.

  L'utilizzo dell'SDK con Apache Maven è stato aggiornato con informazioni su come includere la distinta base dei materiali (BOM) dell'SDK nel progetto.

4 agosto 2015

- I requisiti dei certificati SSL sono un nuovo argomento della sezione Guida introduttiva che descrive il AWS«passaggio ai certificati SHA256 firmati» per le connessioni SSL e come correggere l'utilizzo di questi certificati negli ambienti Java 1.6 e precedenti, necessari per l' AWS accesso dopo il 30 settembre 2015.

  > **Note**
  >
  > Java 1.7+ è già in grado di funzionare con certificati firmati. SHA256

14 maggio 2014

- Il materiale introduttivo e introduttivo è stato ampiamente rivisto per supportare la nuova struttura delle guide e ora include indicazioni su come impostare AWS le credenziali e la regione per lo sviluppo.

  La discussione sugli esempi di codice è stata spostata in un argomento a sé stante nella sezione Documentazione e risorse aggiuntive.

Le informazioni su come visualizzare la cronologia delle revisioni dell'SDK sono state spostate nell'introduzione.

9 maggio 2014

- La struttura generale della AWS SDK per Java documentazione è stata semplificata e gli argomenti Guida introduttiva e Documentazione e risorse aggiuntive sono stati aggiornati.

  Sono stati aggiunti nuovi argomenti:

  - Utilizzo AWS delle credenziali: illustra i vari modi in cui è possibile specificare le credenziali da utilizzare con. AWS SDK per Java
  - Using IAM Roles to Grant Access to AWS Resources on Amazon EC2: fornisce informazioni su come specificare in modo sicuro le credenziali per le applicazioni in esecuzione su istanze. EC2

9 settembre 2013

- Questo argomento, Document History, tiene traccia delle modifiche alla AWS SDK per Java Developer Guide. È inteso come integrazione della cronologia delle note di rilascio.