Guida per gli sviluppatori per la versione 1.x

AWS SDK for Java 1.x

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

Table of Contents

Abbiamo annunciato l'imminente versione end-of-support di AWS SDK for Java (v1). Ti consigliamo di migrare alla AWS SDK for Java v2. Per date, dettagli aggiuntivi e informazioni su come effettuare la migrazione, consulta l'annuncio collegato.

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 for Java 1.x

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

Rilasciata la versione 2 dell'SDK

Dai un'occhiata al nuovoAWS SDK for 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 gli sviluppatoriAWS SDK for Java 2.x.

Documentazione e risorse aggiuntive

Oltre a questa guida, di seguito sono elencate altre risorse online preziose per sviluppatori AWS SDK for Java.

- AWS SDK for Java Documentazione di riferimento delle 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 si sviluppa codice utilizzando l'IDE Eclipse, è possibile utilizzarlo AWS Toolkit for Eclipseper aggiungerloAWS SDK for Java a un progetto Eclipse esistente o per creare un nuovoAWS SDK for Java progetto. Il toolkit supporta anche la creazione e il caricamento diLambda funzioni, l'avvio e il monitoraggio diAmazon EC2 istanze, la gestione diIAM utenti e gruppi di sicurezza, un editor diAWS CloudFormation modelli e altro ancora.

Consulta la Guida perAWS Toolkit for Eclipse l'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 (AWSMobile SDK for Android).

Visualizzazione della cronologia delle revisioni dell'SDK

Per visualizzare la cronologia delle versioni di SDKAWS SDK for Java, incluse le modifiche e i servizi supportati, consulta le note di rilascio dell'SDK.

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

L'AWS SDK for JavaAPI Reference rappresenta la build più recente della versione 1.x dell'SDK. Se stai utilizzando una build precedente della versione 1.x, potresti voler accedere alla documentazione di riferimento SDK che corrisponde 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 suGitHub.
2. Scegli il link`zip` (la maggior parte delle piattaforme, incluso Windows) `otar.gz` (Linux, macOS o Unix) per scaricare l'SDK sul tuo computer.
3. Estrai l'archivio in una directory locale.

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

```
mvn javadoc:javadoc
```

5. Una volta completata la compilazione, 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 for Java.

Argomenti

Configurazione di base con cui lavorare Servizi AWS

Panoramica

Per sviluppare con successo applicazioni a cui accedere Servizi AWS utilizzandoAWS SDK for Java, sono necessarie le seguenti condizioni:

- Devi essere in grado di accedere al portale di AWS accesso disponibile inAWS IAM Identity Center.
- Le autorizzazioni del ruolo IAM configurato per l'SDK devono consentire l'accesso a quanto richiesto dall'Servizi AWSapplicazione. Le autorizzazioni associate alla politica PowerUserAccessAWSgestita sono sufficienti per la maggior parte delle esigenze di sviluppo.
- Un ambiente di sviluppo con i seguenti elementi:
  - File di configurazione condivisi impostati nel modo seguente:
    - Il `config` file contiene un profilo predefinito che specifica unRegione AWS.
    - Il `credentials` file contiene credenziali temporanee come parte di un profilo predefinito.
  - Un'installazione adeguata di Java.
  - Uno strumento di automazione delle build 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 integrare Kit di strumenti AWS s per lavorare più facilmenteServizi AWS. Gli Kit di strumenti AWS per IntelliJe AWS Toolkit for Eclipsesono due toolkit che puoi usare per lo sviluppo in Java.

> **Important**
>
> Le istruzioni in questa sezione di configurazione presuppongono che tu o l'organizzazione utilizziate 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 per l'SDK for Java. Segui queste istruzioni per aggiungere credenziali temporanee al `~/.aws/credentials` file.
> 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 AWS accesso

Il portale di AWS accesso è la posizione Web in cui si accede manualmente all'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 agli AWS SDK e agli strumenti. Non seguite il passaggio 2 perché la AWS SDK for Java versione 1.x non supporta l'aggiornamento automatico dei token e il recupero automatico delle credenziali temporanee per l'SDK descritto nella Fase 2.

Configurare file di configurazione condivisi

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

Configurazione del **`config`** file condiviso

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

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

Per scopi di sviluppo, usa il codice Regione AWS più vicino al punto in cui intendi 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 in questa sezione per creare il `config` file.

Configurazione delle credenziali temporanee per l'SDK

Dopo aver ottenuto l'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 incolli 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. Salva il file. Il file `~/.aws/credentials` dovrebbe ora esistere nel tuo sistema di sviluppo locale. Questo file contiene il profilo [predefinito] utilizzato dall'SDK for Java se non è specificato un profilo con nome specifico.
4. Accedi al portale di AWS accesso.
5. Segui queste istruzioni sotto l'intestazione Aggiornamento manuale delle credenziali per copiare le credenziali del ruolo IAM dal portale di accesso. AWS

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

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

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

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

7. Salva il `credentials` file

L'SDK for Java accederà a queste credenziali temporanee quando crea un client di servizio e le utilizza per ogni richiesta. Le impostazioni per il ruolo IAM scelto nella fase 5a determinano per quanto tempo le credenziali temporanee sono valide. La durata massima è di dodici ore.

Dopo la scadenza delle credenziali temporanee, ripetere le fasi da 4 a 7.

Installare un ambiente di sviluppo Java

AWS SDK for Javarichiede J2SE Development Kit 6.0 o versioni successive. Puoi scaricare la versione più recente del software Java da http://www.oracle.com/technetwork/java/javase/downloads/.

> **Important**
>
> La versione 1.6 di Java (JS2E 6.0) non disponeva del supporto integrato per i certificati SSL con firma SHA256, necessari per tutte le connessioni HTTPS con data successiva al 30 settembre 2015. AWS
> Le versioni Java 1.7 o successive sono dotate di certificati aggiornati e non sono interessate da questo problema.

Selezione di una JVM

Per prestazioni ottimali delle applicazioni basate su server con AWS SDK for Java, ti consigliamo di utilizzare la versione a 64 bit di JVM (Java Virtual Machine). Questa JVM viene eseguita solo in modalità server, anche se durante il runtime si specifica l'opzione `-Client`.

L'utilizzo della versione a 32 bit della JVM con l'`-Server`opzione in fase di esecuzione dovrebbe fornire prestazioni paragonabili alla JVM a 64 bit.

Modi per ottenere il AWS SDK for Java

Prerequisiti

Per utilizzare il AWS SDK for 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 https://sdk-for-java.amazonwebservices.com/latest/ .zip. 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 for 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 for 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 strumenti di creazione

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

Argomenti

Utilizzare gli SDK con Apache Maven

Puoi utilizzare Apache Maven per configurare e creare progetti AWS SDK for Java o 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.

Creare 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 il namespace 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 di avvio rapido, 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`. Ad esempio:

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

> **Note**
>
> Molte più informazioni sulla creazione e la configurazione dei progetti sono fornite nella Guida introduttiva di Maven.

Configura l'SDK come dipendenza Maven

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

Specificazione dei singoli moduli SDK

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

Per utilizzare la 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 delAWS SDK for Java BOM disponibile su Maven Central, visita: https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-bom. Puoi anche usare questa pagina per vedere quali moduli (dipendenze) sono gestiti dalla BOM che puoi includere nella`<dependencies>` sezione del`pom.xml` file del tuo progetto.

Ora puoi selezionare singoli moduli dall'SDK che usi nell'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 AWS Code Sample Catalogper sapere quali dipendenze usare per un determinato datoServizio AWS. Fare riferimento al file POM in un esempio di servizio specifico. Ad esempio, se sei interessato alle dipendenze del servizioAWS S3, consulta l'esempio completo su GitHub. (Guarda il pom in /java/example_code/s3).

Importazione di tutti i moduli SDK

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

```
<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`package` comando di Maven:

```
mvn package
```

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

Crea l'SDK con Maven

Puoi utilizzare Apache Maven per creare l'SDK dai sorgente. Per farlo, scarica il codice SDK da GitHub, decomprimilo localmente ed esegui il seguente comando Maven:

```
mvn clean install
```

Utilizzare gli SDK con Gradle

Per gestire le dipendenze SDK per il tuoGradleprogetto, importa la BOM Maven perAWS SDK for Javanell'applicazione`build.gradle`fascicolo.

> **Note**
>
> Negli esempi seguenti, sostituisci*1.12.529*nel file di build con una versione valida diAWS SDK for Java. Trova la versione più recente nelArchivio centrale Maven.

Configurazione del progetto per Gradle 4.6 o versioni successive

Da Gradle 4.6, puoi 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 ilIMPROVED_POM_SUPPORTfunzionalità in`settings.gradle`file.

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

2. Aggiungi il BOM aldipendenzesezione dell'applicazione`build.gradle`fascicolo.

```
...
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 perAmazon 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 file `build.gradle` 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, sostituisci la dipendenza perAmazon S3con le dipendenze diAWSservizi che utilizzerai nel tuo progetto. I moduli (dipendenze) gestiti daAWS SDK for JavaI BOM sono elencati inArchivio centrale 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. Da gestireAWS SDK for Javale dipendenze per il tuo progetto, usa Spring'splugin per la gestione delle dipendenzeper Gradle per importare la BOM Maven per l'SDK.

1. Aggiungi il plug-in di gestione delle dipendenze alla tua applicazione`build.gradle`file.

```
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 neldipendenzesezione. 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 file `build.gradle` 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, sostituisci la dipendenza perAmazon S3con le dipendenze diAWSservizio che utilizzerai nel tuo progetto. I moduli (dipendenze) gestiti daAWS SDK for JavaI BOM sono elencati inArchivio centrale Maven.

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

Imposta credenzialiAWS temporanee eRegione AWS per lo sviluppo

Per connettersi a uno qualsiasi dei servizi supportati con ilAWS SDK for Java, è necessario fornire credenzialiAWS temporanee. GliAWS SDK e le CLI utilizzano le catene di provider per cercare le credenzialiAWS temporanee in luoghi diversi, tra cui le variabili di ambiente utente o di sistema e i file diAWS configurazione locali.

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

- Quando usi un'istanza EC2, crea un ruolo IAM e quindi consenti alla tua istanza EC2 di accedere a quel ruolo, come mostrato in Utilizzo dei ruoli IAM per concedere l'accesso alleAWS risorse suAmazon EC2.
- ConfiguraAWS le credenziali all'interno di Eclipse utilizzando il AWS Toolkit for Eclipse. Per ulteriori informazioni, consulta Configurazione delleAWS credenziali nella Guida perAWS Toolkit for Eclipse l'utente.

Configurazione di credenziali temporanee

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

- Imposta le credenziali temporanee nel file del profiloAWS delle credenziali sul tuo sistema locale, situato in:

- `~/.aws/credentials` su Linux, macOS o Unix
- `C:\Users\USERNAME\.aws\credentials` in Windows

Consultathe section called “Configurazione delle credenziali temporanee per l'SDK” questa guida per istruzioni su come ottenere le tue credenziali temporanee.

- Imposta le variabili`AWS_ACCESS_KEY_ID`, `AWS_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
  ```

- Per un'istanza di EC2, specificare un ruolo IAM e concedere all'istanza EC2 l'accesso a tale ruolo. Per una discussione dettagliata su come funziona, consulta IAM Roles forAmazon EC2 nellaAmazon EC2 User Guide for Linux Instances.

Dopo aver impostato le credenzialiAWS temporanee utilizzando uno di questi metodi, verranno caricate automaticamenteAWS SDK for Java utilizzando la catena di fornitori di credenziali predefinita. Per ulteriori informazioni sull'utilizzoAWS delle credenziali nelle applicazioni Java, vedere UtilizzoAWS delle credenziali.

Aggiornamento delle credenziali IMDS

AWS SDK for Java supporta la scelta di aggiornare le credenziali IMDS in background ogni minuto, indipendentemente dal tempo di scadenza delle credenziali. Ciò consente di aggiornare le credenziali più frequentemente e riduce la possibilità che il mancato raggiungimento dell'IMDS influisca sullaAWS disponibilità percepita.

```
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 ilRegione AWS

È necessario impostare un valore predefinitoRegione AWS che verrà utilizzato per accedere aiAWS servizi conAWS SDK for Java. Per ottimizzare le prestazioni di rete, scegliere una regione geograficamente vicina alla propria posizione (o ai clienti). Per un elenco delle regioni per ciascun servizio, vedere Regioni ed endpoint nella Guida di riferimentoAmazon Web Services generale.

> **Note**
>
> Se non si seleziona una regione, di default verrà utilizzato us-east-1.

Puoi utilizzare tecniche simili per impostare le credenziali per impostare laAWS regione predefinita:

- Imposta il fileRegione AWS nel file diAWS 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 file desideratoRegione AWS (ad esempio, «us-east-1") con your_aws_region.

- Imposta la variabile di ambiente `AWS_REGION`.

  In 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 ilRegione AWS nome desiderato.

Utilizzando il AWS SDK for Java

Questa sezione fornisce importanti informazioni generali sulla programmazione e si applicano a tutti i servizi AWS SDK for 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 for Java Esempi di codice.

Argomenti

Migliori pratiche per AWS lo sviluppo con AWS SDK for Java

Le seguenti best practice possono aiutarti a evitare problemi o problemi durante lo sviluppo di AWS applicazioni con. AWS SDK for 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 for 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 consigliata

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 for 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, l'interfaccia di servizio per DynamoDB è denominata dbClient. AmazonDynamo Ogni interfaccia di servizio ha un client builder corrispondente che è possibile utilizzare per costruire un'implementazione dell'interfaccia di servizio. La classe client builder for DynamoDB è denominata DB. AmazonDynamo ClientBuilder

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 for 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

Entrambi i costruttori di client sync e async 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 di una propria risorsa di 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 for Java

Per effettuare richieste a Amazon Web Services, è necessario fornire credenziali AWS temporanee AWS SDK for 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 for Java tenta di trovare credenziali temporanee utilizzando la catena di fornitori di credenziali predefinita implementata dalla classe Default. AWSCredentialsProviderChain La catena di provider delle credenziali predefinita cerca le credenziali in questo ordine:

1. Variabili di ambiente -`AWS_ACCESS_KEY_ID`, e. `AWS_SECRET_ACCESS_KEY` `AWS_SESSION_TOKEN` AWS SDK for Java utilizza la EnvironmentVariableCredentialsProviderclasse per caricare queste credenziali.
2. Proprietà del sistema Java - `aws.accessKeyIdaws.secretKey`, e`aws.sessionToken`. li AWS SDK for Java utilizza SystemPropertiesCredentialsProviderper caricare queste credenziali.
3. Credenziali Web Identity Token dall'ambiente o dal contenitore.
4. Il file dei profili di credenziali predefinito, generalmente situato in `~/.aws/credentials` (la posizione può variare in base alla piattaforma) e condiviso da molti AWS SDK e da. AWS CLI Lo AWS SDK for Java utilizza per ProfileCredentialsProvidercaricare 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 for Java utilizza per caricare queste ContainerCredentialsProvidercredenziali. È possibile specificare l'indirizzo IP per questo valore.
6. Credenziali del profilo di istanza: utilizzate sulle istanze EC2 e fornite tramite il Amazon EC2 servizio di metadati. Le AWS SDK for 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 EC2 ContainerCredentialsProviderWrapper.

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 “Configurazione di 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 istanze EC2, consulta Using IAM Roles to Grant Access to Resources on. AWS Amazon EC2

Imposta un profilo di credenziali alternativo

AWS SDK for 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 AWS SDK e gli strumenti ufficialmente supportati (inclusi gli AWS CLI e i AWS Tools for Windows PowerShell). Per modificare solo il profilo di un'applicazione Java, potete invece utilizzare la proprietà `aws.profile` di sistema.

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

Imposta una posizione alternativa per il file delle credenziali

AWS SDK for 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 for 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 e`metadata_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 fornitore di credenziali o di una catena di provider a un client builder che accetta un'AWSCredentialsProviderinterfaccia 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 for Java forniti, consulta All Known Implementation Classes in. AWSCredentialsProvider

> **Note**
>
> È possibile utilizzare questa tecnica per fornire fornitori 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 oggetto AWSStaticCredentialsProvider. `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 for 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 in modo che utilizzi la regione Europa (Irlanda), utilizza 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 provider delle credenziali/regioni predefinita per determinare la regione dell'ambiente, utilizzare il metodo `defaultClient` del generatore client.

```
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 for 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 for 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 for 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 restituirà 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 correttamente 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 for 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 for Java Fornisce due modi: oggetti futuri e metodi di callback.

Java Futures

I metodi asincroni AWS SDK for 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 che blocca i 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 for 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 for 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 for Java Registrazione delle chiamate

AWS SDK for 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 for 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 Gli ID di richiesta sono accessibili a livello di codice tramite gli oggetti Exception nell'SDK per qualsiasi chiamata di servizio fallita e possono anche essere segnalati tramite il livello di registro DEBUG nel logger «com.amazonaws.request».

Il seguente file log4j.properties consente un riepilogo delle richieste e delle risposte, inclusi gli ID delle richieste. AWS

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

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

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 for 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 for Java

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

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 for 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 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 for 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 for 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 for 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 esempio 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 for Java

AWS SDK for 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 delle tue JVM se utilizzate 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 for 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. Per 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 for 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}
```

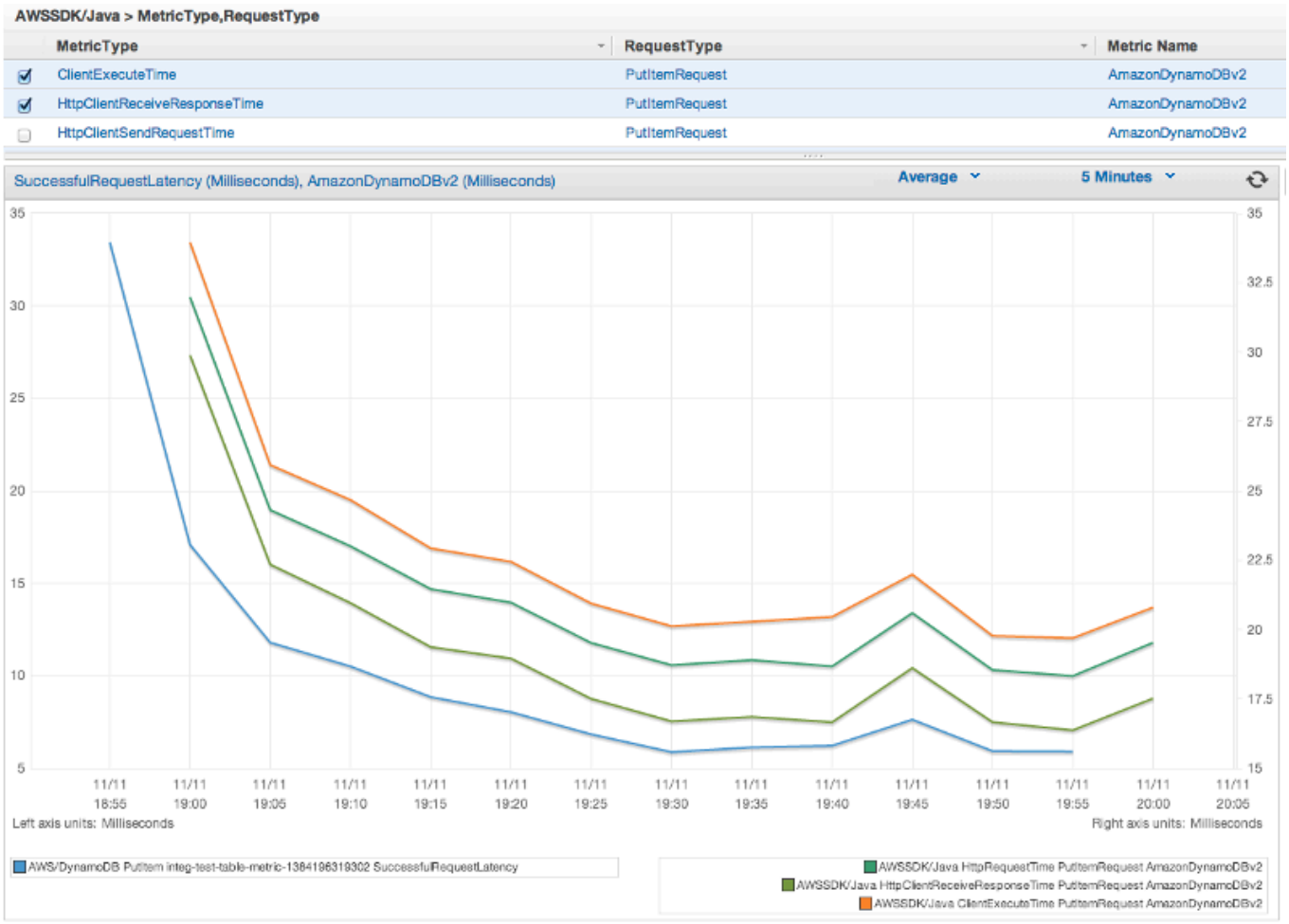
Una volta abilitata la funzionalità, ogni volta che viene inviata una richiesta di servizio, i dati metrici verranno generati AWS SDK for Java, messi in coda per il riepilogo statistico e caricati in modo asincrono circa una volta al minuto. AWS CloudWatch Una volta caricate le metriche, puoi visualizzarle utilizzando AWS Management Consolee 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:

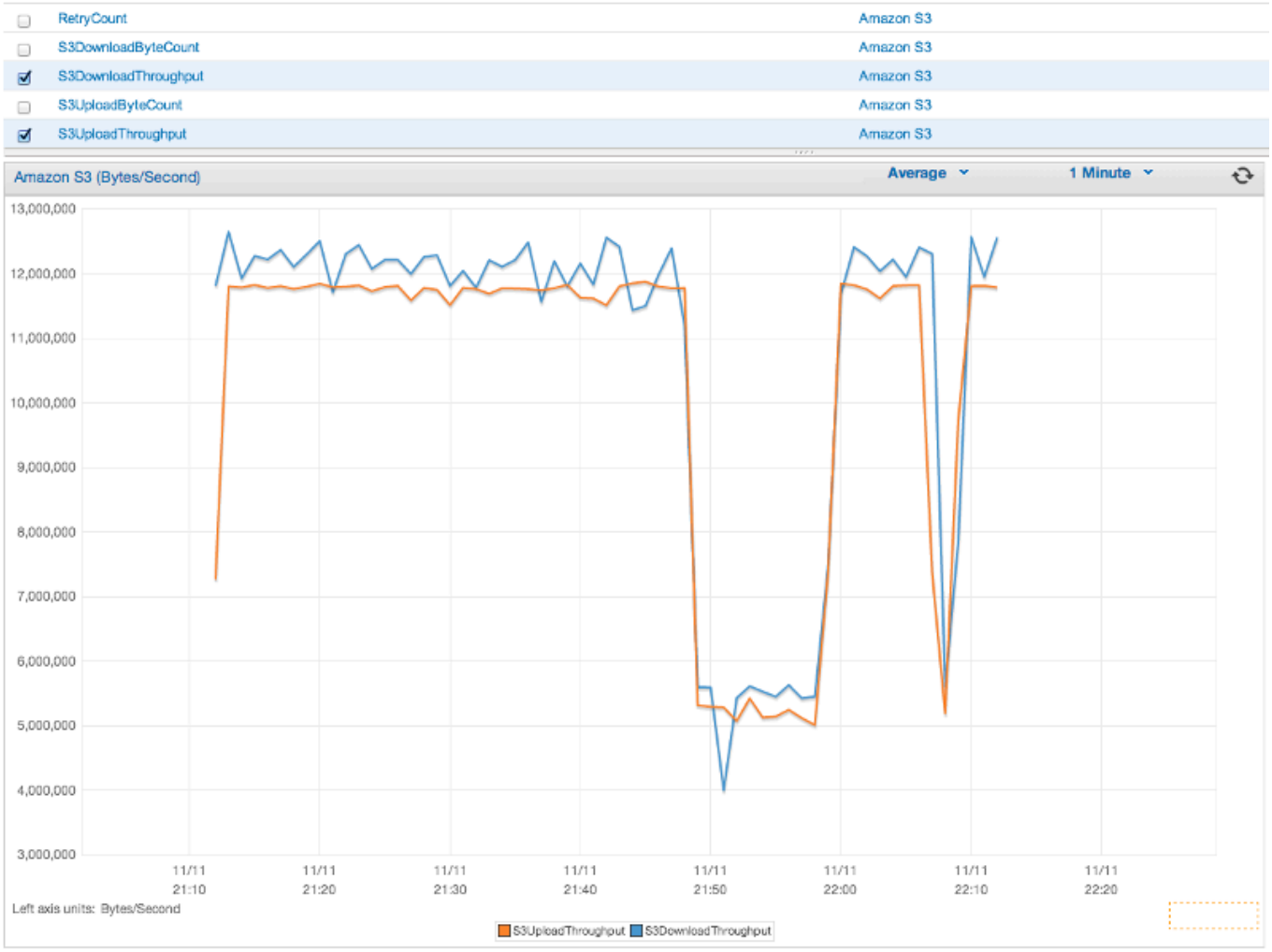
AWS Richiedi metriche

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



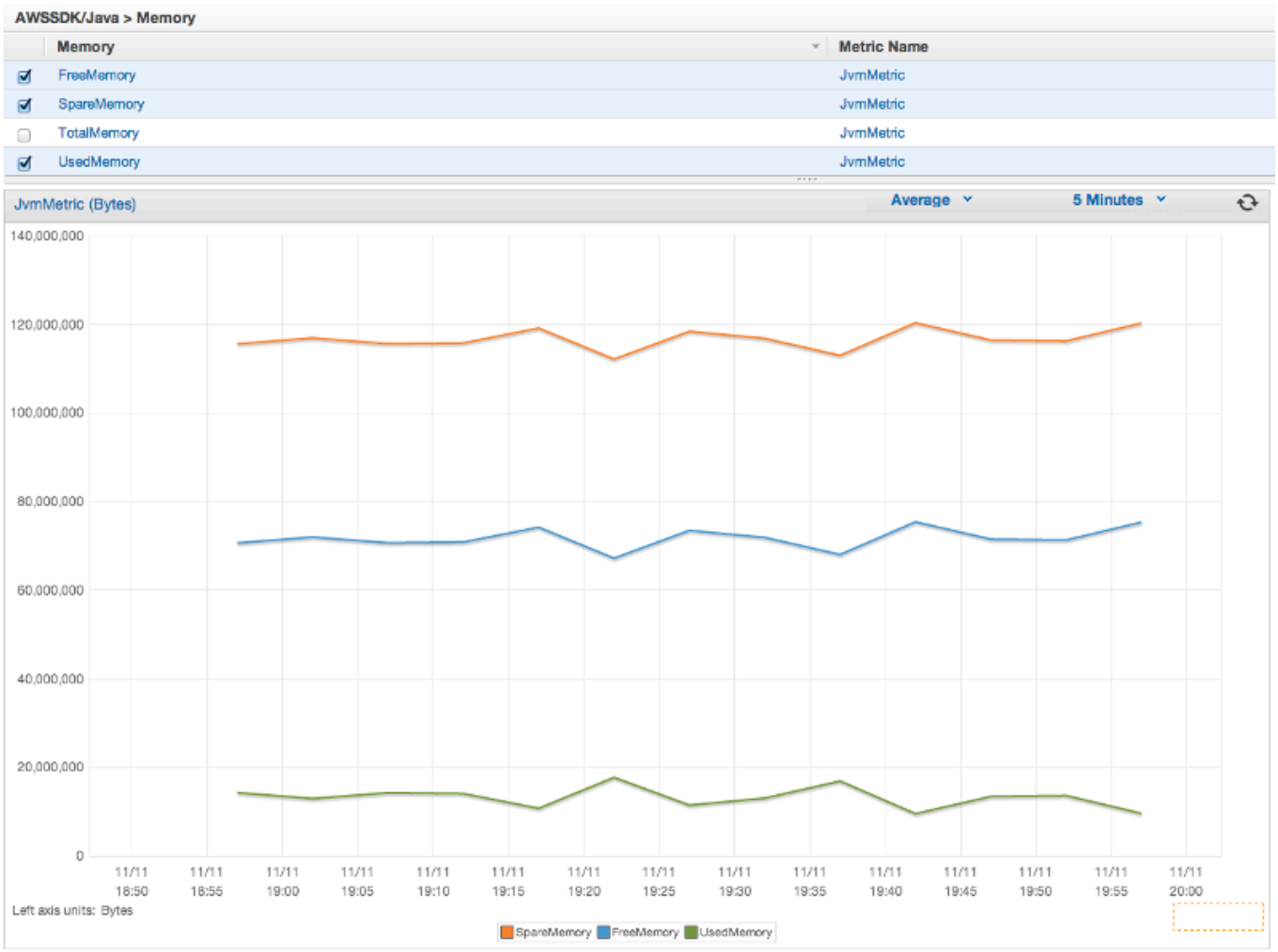
Servizio AWS Metriche

- 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 for Java CloudWatch AWS SDK for Java

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

Esempi di codice dell'AWS SDK for Java

Questa sezione fornisce tutorial ed esempi di utilizzo dellaAWS SDK for Java versione 1 per programmareAWS i servizi.

Trova il codice sorgente di questi e altri esempi nel repository degli esempi di codice dellaAWS documentazione suGitHub.

Per proporre un nuovo esempio di codice per il team della documentazione AWS che consideri la produzione, 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 la contribuzione nel repository degli esempi di codice suGitHub..

AWS SDK for Java2.x

Nel 2018,AWS ha rilasciato il AWS SDK for Java 2.x. Questa guida contiene istruzioni sull'uso della versione più recente di Java SDK insieme a un codice di esempio.

> **Note**
>
> Consulta Documentazione e risorse aggiuntive per ulteriori esempi e risorse aggiuntive disponibili perAWS SDK for Java gli sviluppatori!

Esempi di CloudWatch che utilizzano ilAWS SDK for Java

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

Amazon CloudWatch monitora iAmazon Web Services(AWS) risorse e le applicazioni eseguite suAWSin tempo reale. È possibile utilizzare CloudWatch per raccogliere e tenere traccia dei parametri, che sono delle variabili che si possono misurare per le risorse e le applicazioni. CloudWatch Gli allarmi di inviano notifiche o apportano automaticamente modifiche alle risorse monitorate in base alle regole definite.

Per ulteriori informazioni su CloudWatch, consulta la Guida per l'utente di Amazon CloudWatch.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ciascuna 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

Recupero dei parametri da CloudWatch

Elencazione dei parametri

Per elencare CloudWatch metriche, crea unListMetricsRequest dell'elencoe chiama AmazonCloudWatchClient`listMetrics`metodo. 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 che vengono pubblicate daAWSi servizi sono disponibili all'interno del {https://docs-aws-amazon-com-AmazonCloudWatch-latest-monitoring-CW-support-for-AWS-html} [Amazon] CloudWatch Informazioni di riferimento su dimensioni e parametri] nelAmazon CloudWatchGuida per l'utente di .

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;
    }
}
```

I parametri vengono restituiti in un oggettoListMetricsResultchiamando il suo`getMetrics`metodo. I risultati possono essere paginati. Per recuperare il batch successivo di risultati, chiama`resetNextToken`sull'oggetto richiesta originale con il valore restituito del`ListMetricsResult`dell'oggetto`getNextToken`metodo e ripassa l'oggetto richiesta modificato a un'altra chiamata `alistMetrics`.

Ulteriori informazioni

- ListMetricsnellaAmazon CloudWatchInformazioni di riferimento sull'API.

Pubblicazione di dati dei parametri personalizzati

DiversiAWSservizi pubblicanometriche proprienegli spazi dei nomi che iniziano con»`AWS` «Puoi anche pubblicare i dati dei parametri personalizzati utilizzando il tuo spazio dei nomi (purché non inizi con»`AWS` «).

Pubblicare dati dei parametri personalizzati

Per pubblicare i dati dei parametri, chiama AmazonCloudWatchClient`putMetricData`metodo con un`PutMetricDataRequest`. `PutMetricDataRequest` deve includere lo spazio dei nomi personalizzato da utilizzare per i dati e le informazioni sul punto dati stesso in un oggetto `MetricDatum`.

> **Note**
>
> Non puoi specificare uno spazio dei nomi che inizia con».AWS «. Namespace che iniziano con»AWS «sono riservati per l'utilizzo da parte diAmazon Web ServicesProdotti .

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 diAmazon CloudWatchParametrinellaAmazon CloudWatchGuida per l'utente di .
- AWSSpazi dei nominellaAmazon CloudWatchGuida per l'utente di .
- PutMetricDatanellaAmazon CloudWatchInformazioni di riferimento sull'API.

Utilizzo di CloudWatch Allarmi

Creazione di un allarme

Per creare un allarme basato su un CloudWatch metrica, chiama AmazonCloudWatchClient`putMetricAlarm`con un metodoPutMetricAlarmRequestriempito con 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 il valore CloudWatch allarmi creati, chiama il client di AmazonCloudWatchClient`describeAlarms`con un metodoDescribeAlarmsRequestche è possibile utilizzare per impostare le opzioni per il 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 di allarmi può essere ottenuto chiamandogetMetricAlarmssulDescribeAlarmsResult allarmiviene restituito dadescribeAlarms.

I risultati possono essere paginati. Per recuperare il batch successivo di risultati, chiamasetNextTokensull'oggetto richiesta originale con il valore restituito delDescribeAlarmsResultoggettogetNextTokenmetodo e ripassa l'oggetto richiesta modificato a un'altra chiamata adescribeAlarms.

> **Note**
>
> Puoi anche recuperare allarmi per un parametro specifico utilizzando AmazonCloudWatchClientdescribeAlarmsForMetricmetodo. L'uso è simile a describeAlarms.

Elimina allarmi

Per eliminare CloudWatch allarmi, chiama AmazonCloudWatchClientdeleteAlarmscon un metodoDeleteAlarmsRequestcontenente uno o più nomi di allarmi da 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 diAmazon CloudWatchAllarminellaAmazon CloudWatchGuida per l'utente di
- PutMetricAlarmnellaAmazon CloudWatchDocumentazione di riferimento API
- DescribeAlarmsnellaAmazon CloudWatchDocumentazione di riferimento API
- DeleteAlarmsnellaAmazon CloudWatchDocumentazione di riferimento API

Utilizzo di operazioni di allarme in CloudWatch

Utilizzo di CloudWatch operazioni di allarme di, puoi creare allarmi che eseguono operazioni come l'arresto automatico, la terminazione, il riavvio o il ripristinoAmazon EC2istanze.

> **Note**
>
> Operazioni di allarme possono essere aggiunte a un allarme utilizzando il metodo di PutMetricAlarmRequest`setAlarmActions` durante la creazione di un allarme.

Attivare le operazioni di allarme

Per abilitare le azioni di allarme per un CloudWatch allarme, chiama AmazonCloudWatchClient`enableAlarmActions`con unEnableAlarmActionsRequestcontenente uno o più nomi di allarmi di cui si desidera attivare le operazioni.

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 allarme per un CloudWatch allarme, chiama AmazonCloudWatchClient`disableAlarmActions`con unDisableAlarmActionsRequestcontenente uno o più nomi di allarmi di cui si desidera disabilitare le operazioni.

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

- Creazione di allarmi per arrestare, terminare, riavviare o recuperare un'istanzanellaAmazon CloudWatchGuida per l'utente di
- PutMetricAlarmnellaAmazon CloudWatchDocumentazione di riferimento API
- EnableAlarmActionsnellaAmazon CloudWatchDocumentazione di riferimento API
- DisableAlarmActionsnellaAmazon CloudWatchDocumentazione di riferimento API

Invio di eventi ad CloudWatch

CloudWatchEvents offre un flusso quasi in tempo reale di eventi di sistema che descrivono le modifiche inAWSrisorse perAmazon EC2istanze,Lambdafunzioni,Kinesisflussi,Amazon ECSattività,Step Functionsmacchine a stati,Amazon SNSargomenti,Amazon SQScode o destinazioni incorporate. Puoi abbinare gli eventi e instradarli verso una o più funzioni o stream target utilizzando regole semplici.

Aggiunta di eventi

Per aggiungere personalizzatiCloudWatcheventi, chiama Amazon CloudWatcheventsClient`putEvents`metodo con unPutEventsRequestoggetto che contiene uno o piùPutEventsRequestEntryoggetti 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, chiama il client di AmazonCloudWatchEventsClient`putRule`emetodo con unPutRuleRequest inserimentocon il nome della regola e parametri opzionali come ilModello di eventi,IAMruolo da associare alla regola e a espressione di pianificazioneche descrive la frequenza con cui viene eseguita la 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. Esempi di target includono istanze Amazon EC2, funzioni Lambda, stream Kinesis, operazioni Amazon ECS, macchine a stati di Step Functions e target integrati.

Per aggiungere un target a una regola, chiama il client di AmazonCloudWatchEventsClient`putTargets`metodo con unPutTargetsRequest Puttcontenente la regola da aggiornare e un elenco di target 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 con PutEventsnellaAmazon CloudWatch EventsGuida per l'utente di
- Pianificazione di espressioni per regolenellaAmazon CloudWatch EventsGuida per l'utente di
- Tipi di eventi perCloudWatchEventinellaAmazon CloudWatch EventsGuida per l'utente di
- Eventi e modelli di eventinellaAmazon CloudWatch EventsGuida per l'utente di
- PutEventsnellaAmazon CloudWatch EventsDocumentazione di riferimento API
- PutTargetsnellaAmazon CloudWatch EventsDocumentazione di riferimento API
- PutRulenellaAmazon CloudWatch EventsDocumentazione di riferimento API

Esempi di DynamoDB utilizzando l'AWS SDK for Java

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

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ciascuna 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 tabelle inDynamoDB

Le tabelle sono container per tutti gli item in un database DynamoDB. Prima di poter aggiungere o rimuovere dati da DynamoDB, devi 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 ScalarAttributeTypeclasse. 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 gliAmazon DynamoDB sviluppatori.
- Valori di velocità effettiva forniti che definiscono il numero di unità di capacità di lettura/scrittura riservate per la tabella.

> **Note**
>
> Amazon DynamoDBi prezzi si basano sui valori di produttività forniti che hai impostato sui tuoi tavoli, quindi prenota solo la capacità che ritieni necessaria per il tuo tavolo.

La produttività fornita per una tabella può essere modificata in qualsiasi momento, in modo da poter regolare la capacità in base alle esigenze.

Creazione di una tabella

Usa il`createTable` metodo del DynamoDBclient per creare una nuovaDynamoDB 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 dellaDynamoDB tabella.

> **Note**
>
> Se esiste già una tabella con il nome scelto, 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 suGitHub.

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);
```

Vedi l'esempio completo suGitHub.

Elencare tabelle

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

> **Note**
>
> Se la tabella denominata non esiste per il tuo account e la tua regione, ResourceNotFoundExceptionviene 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.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: utilizzale`getLastEvaluatedTableName` sull'ListTablesResultoggetto restituito per ottenere l'ultima tabella che è stata 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 suGitHub.

Descrizione di (recupero delle informazioni su) una tabella

Chiama il`describeTable` metodo del DynamoDBcliente.

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

Modifica (aggiornamento) di una tabella

È possibile modificare i valori di throughput forniti dalla tabella in qualsiasi momento chiamando il`updateTable` metodo del DynamoDBclient.

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

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 suGitHub.

Eliminazione di una tabella

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

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

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 suGitHub.

Ulteriori informazioni

- Linee guida per l'utilizzo delle tabelle nella Guida per gliAmazon DynamoDB sviluppatori
- Utilizzo delle tabelleDynamoDB nella Guida per gliAmazon DynamoDB sviluppatori

Uso di item inDynamoDB

In DynamoDB, una voce è una raccolta di attributi, ognuno dei quali dispone di un nome e un valore. Un valore attributo può essere un tipo scalare, set o documento. Per ulteriori informazioni, consultaRegole di denominazione e tipi di datinellaAmazon DynamoDBGuida per sviluppatori di .

Recuperare (ottenere) un item da una tabella

Chiama l'AmazonDynamoDB`getItem`metodo e passalo aGetItemRequestoggetto con il nome della tabella e il valore della chiave primaria dell'item desiderato. Restituisce unGetItemResultoggetto.

Puoi utilizzare il metodo `getItem()` dell'oggetto `GetItemResult` restituito per recuperare una mappa di coppie chiave (String) e valore (AttributeValue) associate alla voce.

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 l'account e la regione, viene generata un'eccezione ResourceNotFoundException.

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

Puoi aggiornare un attributo per un item che esiste già in una tabella utilizzando AmazonDynamoDB`updateItem`fornisce un nome di tabella, un valore chiave primaria e una mappa di campi da aggiornare.

> **Note**
>
> Se la tabella denominata non esiste per l'account e la regione, oppure se la voce identificata dalla chiave primaria passata non esiste, viene generata un'eccezione ResourceNotFoundException.

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.

Utilizza la classe DynamoDBMapper

LaAWS SDK for Javafornisce unMapper DynamoDBMapperclass, che ti permette di mappare le classi lato clientAmazon DynamoDBtavoli. Per utilizzare il pluginMapper DynamoDBMapperclass, definisci la relazione tra gli item in unaDynamoDBtable e le istanze di oggetto corrispondenti nel codice utilizzando annotazioni (come mostrato nell'esempio di codice seguente). LaMapper DynamoDBMapperclass ti consente di accedere alle tabelle, eseguire diverse operazioni di creazione, lettura, aggiornamento ed eliminazione (CRUD) ed eseguire query.

> **Note**
>
> LaMapper DynamoDBMapperclass non ti 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 illustra come aggiungere contenuto allaMusicatabella utilizzando ilMapper DynamoDBMapperclasse. Dopo aver aggiunto il contenuto alla tabella, si noti che un elemento viene caricato utilizzando ilPartizioneeOrdinachiavi. Poi ilPremil'articolo è aggiornato. Per informazioni sulla creazione delMusicatabella, vediCreazione di una tabellanellaAmazon DynamoDBGuida per sviluppatori di .

```
        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 di itemnellaAmazon DynamoDBGuida per gli sviluppatori
- Uso di item inDynamoDBnellaAmazon DynamoDBGuida per gli sviluppatori

Esempi di Amazon EC2 utilizzando l'AWS SDK for Java

In questa sezione vengono forniti alcuni esempi di programmazioneAmazon EC2con ilAWS SDK for Java.

Argomenti

- Tutorial: Avvio di un'istanza EC2
- Utilizzo di ruoli IAM per concedere l'accesso aAWSRisorse suAmazon EC2
- Tutorial: Amazon EC2 Spot Instances
- Tutorial: AvanzatoAmazon EC2Gestione delle richieste Spot
- Gestione di istanze Amazon EC2
- Utilizzo di indirizzi IP elastici in Amazon EC2
- Utilizzo di regioni e zone di disponibilità
- Utilizzo di coppie di chiavi Amazon EC2
- Lavorare con i gruppi di sicurezza in Amazon EC2

Tutorial: Avvio di un'istanza EC2

Questa esercitazione mostra come utilizzare laAWS SDK for Javaper avviare un'istanza EC2.

Argomenti

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

- EseguireAmazon EC2Istanza

Prerequisiti

Prima di iniziare, assicurati di aver creato unaAccount AWSe che hai configurato la tuaAWSCredenziali . 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. Suggeriamo di effettuare la migrazione da EC2-Classic a un VPC. Per ulteriori informazioni, consulta Migrare da EC2-Classic a un VPC nella Amazon EC2 User Guide o nellaAmazon EC2 User Guide. Consulta anche 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. È possibile creare un gruppo di sicurezza che consente alle istanze EC2 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 for Java.

È possibile creare un gruppo di sicurezza da utilizzare in EC2-Classic o EC2-VPC. Per ulteriori informazioni su EC2-Classic ed EC2-VPC, consulta Supported Platforms nella Guida per l' Amazon EC2 utente per le istanze Linux.

Per ulteriori informazioni sulla creazione di un gruppo di sicurezza utilizzando la Amazon EC2 console, consulta Amazon EC2 Security Groups nella Guida utente per le Amazon EC2 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 diversi IP, porte e protocolli 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 key pair quando si avvia un'istanza EC2 e una chiave privata della key pair quando ci si connette all'istanza. È possibile creare una key pair oppure utilizzare una key pair esistente utilizzata all'avvio di altre istanze. Per ulteriori informazioni, consultaAmazon EC2Coppie di chiavinellaAmazon EC2Guida per l'utente per istanze Linux.

1. Crea e inizializza unCreateKeyPairRequestistanza. Utilizzo dell'withKeyNamemetodo per impostare il nome della key pair, come segue:

```
CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();

createKeyPairRequest.withKeyName(keyName);
```

> **Important**
>
> I nomi delle coppie di chiavi devono essere univoci. Se si tenta di creare una key pair con lo stesso nome di una key pair esistente, si otterrà un'eccezione.

2. Passa l'oggetto della richiesta alcreateKeyPairmetodo. Il metodo restituisce unCreateKeyPairResultistanza, come segue:

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

3. Chiama l'oggetto risultatogetKeyPairmetodo per ottenere unKeyPairOggetto. Chiama il`KeyPair`oggettogetKeyMaterialper ottenere la chiave privata non crittografata, con codifica PEM, come indicato di seguito:

```
KeyPair keyPair = new KeyPair();

keyPair = createKeyPairResult.getKeyPair();

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

EseguireAmazon EC2Istanza

Utilizza la procedura seguente per avviare una o più istanze EC2 configurate in modo identico dalla stessa Amazon Machine Image (AMI). Dopo aver creato le istanze EC2, puoi controllarne lo stato. Dopo che le istanze EC2 sono in esecuzione, puoi connetterti a esse.

1. Crea e inizializza unRunInstancesRequestistanza. Verifica che l'AMI, la key pair e il gruppo di sicurezza indicati esistano nella regione specificata durante la creazione dell'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 informazioni su come trovare le AMI pubbliche fornite da Amazon o crearne di proprie, consultaAmazon Machine Image (AMI).

withInstanceType

- Un tipo di istanza compatibile con l'AMI specificata. Per ulteriori informazioni, consultaTipi di istanzanellaAmazon EC2Guida per l'utente per istanze Linux.

withMinCount

- Il numero minimo di istanze EC2 da avviare. Se si tratta di più istanze rispetto a quelle che Amazon EC2 può avviare nella zona di disponibilità di destinazione, Amazon EC2 non avvia istanze.

withMaxCount

- Il numero massimo di istanze EC2 da avviare. Se si tratta di più istanze rispetto a quelle che Amazon EC2 può avviare nella zona di disponibilità di destinazione, Amazon EC2 avvia il numero maggiore possibile di istanze superiore a `MinCount`. È possibile avviare tra 1 e il numero massimo di istanze consentite per il tipo di istanza. Per ulteriori informazioni, consulta Quante istanze è possibile eseguire inAmazon EC2nellaAmazon EC2Domande frequenti sui generici

withKeyName

- Il nome della coppia di chiavi EC2. 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, consultaCreazione di unAmazon EC2Gruppo di sicurezza.

2. Avvia le istanze passando l'oggetto richiesta alrunInstancesmetodo. Il metodo restituisce unRunInstancesResultoggetto, come indicato di seguito:

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

Dopo che l'istanza è in esecuzione, puoi connetterti a essa tramite la key pair. Per ulteriori informazioni, consultaConnessione a un'istanza Linux. nelAmazon EC2Guida per l'utente per istanze Linux.

Utilizzo di ruoli IAM per concedere l'accesso aAWSRisorse suAmazon EC2

Tutte le richieste aAmazon Web Services(AWS) deve essere firmato crittograficamente utilizzando le credenziali rilasciate daAWS. È possibile utilizzareRuoli IAMper garantire comodamente un accesso sicuro aAWSrisorse dal tuoAmazon EC2istanze.

In questo argomento vengono fornite le informazioni su come utilizzare i ruoli IAM con applicazioni Java SDK in esecuzione suAmazon EC2. Per ulteriori informazioni sulle istanze IAM, consultaRuoli IAM perAmazon EC2nellaAmazon EC2Guida per l'utente per istanze Linux.

La catena di provider predefinita e i profili di istanza EC2

Se la tua applicazione crea unAWSclient che utilizza il costruttore predefinito, quindi il client cercherà le credenziali utilizzando ilcatena di provider di credenziali predefinite, nel seguente ordine:

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 consegnate tramite ilAmazon EC2servizio container se il servizio container`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`è impostata la variabile di ambiente e security manager ha il permesso di accedere alla variabile.
5. Nelle credenziali del profilo di istanza, disponibili all'interno dei metadati dell'istanza associati al ruolo IAM per l'istanza EC2.
6. Credenziali Web Identity Token dall'ambiente o dal contenitore.

Lacredenziali dell'istanzanella catena di provider predefinita è disponibile solo quando si esegue l'applicazione su unAmazon EC2ma offre la massima semplicità di utilizzo e la massima sicurezza durante l'utilizzo conAmazon EC2istanze. Puoi anche passare un'istanza InstanceProfileCredentialsProvider direttamente al costruttore client per ottenere le credenziali del profilo di istanza senza passare attraverso l'intera catena di provider predefinita.

Ad esempio:

```
AmazonS3 s3 = AmazonS3ClientBuilder.standard()
```

```
            .withCredentials(new InstanceProfileCredentialsProvider(false))
            .build();
```

Quando si utilizza questo approccio, l'SDK recupera temporaneamenteAWScredenziali che hanno le stesse autorizzazioni di quelle associate al ruolo IAM associato al ruolo IAM associato alAmazon EC2Istanza nel suo profilo dell'istanza. Sebbene queste credenziali siano temporanee e alla fine scadano,`InstanceProfileCredentialsProvider`Le aggiorna periodicamente da per l'utente, affinché le credenziali ottenute continuino a consentire l'accesso aAWS.

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

Se il costruttore client non riesce a trovare le credenziali utilizzando la catena del provider di credenziali, verrà generato unAmazonClientException.

Procedura guidata: Utilizzo di ruoli IAM per istanze EC2

Le procedure guidate seguenti illustrano come recuperare un oggetto daAmazon S3utilizzo di un ruolo IAM per gestire l'accesso.

Creare un ruolo IAM

Crea un ruolo IAM che concede accesso in sola lettura a Amazon S3.

1. Aprire la console IAM.
2. Nel riquadro di navigazione selezionareRuoli, quindiCreazione di nuovo ruolo.
3. Inserisci un nome per il ruolo, quindi seleziona Next Step (Fase successiva). Ricorda questo nome, poiché sarà necessario all'avvio del tuoAmazon EC2istanza.
4. SulSeleziona il tipo di ruolopagina, sotto Servizio AWSRuoli, seleziona Amazon EC2 .
5. SulImpostazione delle autorizzazionipagina, sottoSeleziona modello di policy, seleziona Amazon S3Read Only Access (Accesso in sola lettura a), quindiFase successiva.

6. SulReview (Revisione)page, selezionaCreazione di ruolo.

Avvio di un'istanza EC2 e specifica del ruolo IAM

È possibile lanciare unAmazon EC2istanza con un ruolo IAM utilizzando ilAmazon EC2console o ilAWS SDK for Java.

- Per lanciare unAmazon EC2utilizzando la console, segui le indicazioni fornite inNozioni di base suAmazon EC2Istanze LinuxnellaAmazon EC2Guida per l'utente per istanze Linux.

  Quando raggiungi la pagina Review Instance Launch (Verifica del lancio dell'istanza), seleziona Edit instance details (Modifica dettagli istanza). Nello statoRuolo IAM, selezionare il ruolo IAM 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 lanciare unAmazon EC2istanza con un ruolo IAM utilizzando ilAWS SDK for Java, consultaEseguire una diAmazon EC2Istanza.

Crea la tua applicazione

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

Successivamente, copia del fileAWS SDK for Javalibrerie nella tua directory appena creata. Se è stata scaricata laAWS SDK for Javaal tuo`~/Downloads`directory, è possibile copiarle utilizzando i comandi seguenti:

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

Apri un nuovo file, chiamalo`GetS3Object.java`e aggiungere 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 aggiungere 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>
```

Creare ed eseguire il programma modificato. Notare che nel programma non sono presenti credenziali. Pertanto, a meno che tu non abbia il tuoAWScredenziali già specificate, il codice verrà generato`AmazonServiceException`. Ad 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 all'istanza EC2

Trasferisci il programma sul tuoAmazon EC2istanza che utilizza la copia sicura ( ), insieme alAWS SDK for JavaLibrerie . 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,nome utentepotrebbe essere «ec2-user», «root» o «ubuntu». Per ottenere il nome DNS pubblico dell'istanza, aprire laConsole EC2e cerca ilDNS pubblicovalore nellaDescription (Descrizione)tab (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
- lo`libe`e`third-party`le directory sono le cartelle di libreria corrispondenti dalAWS SDK for Java.
- La`-r`l'interruttore indica che`scp`dovrebbe fare una copia ricorsiva di tutti i contenuti del`library`e`third-party`directory nelAWS SDK for Javadistribuzione.
- La`-p`l'interruttore indica che`scp`dovrebbe preservare le autorizzazioni dei file di origine quando li copia nella destinazione.

> **Note**
>
> La`-p`funziona solo su Linux, macOS o Unix. Se si copiano file da Windows, potrebbe essere necessario correggere le autorizzazioni del file sull'istanza utilizzando il seguente comando:

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

Eseguire il programma di esempio sull'istanza EC2

Per eseguire il programma, connettiti al tuoAmazon EC2istanza. Per ulteriori informazioni, consultaConnessione a un'istanza LinuxnellaAmazon EC2Guida per l'utente per istanze Linux.

Se **`ant`** non è disponibile nell'istanza, installarla utilizzando il seguente comando:

```
sudo yum install ant
```

Quindi, esegui il programma utilizzando`ant`come segue:

```
ant run
```

Il programma scriverà il contenuto del tuoAmazon S3passare alla finestra dei comandi.

Tutorial: Amazon EC2 Spot Instances

Panoramica

Le istanze Spot ti consentono di fare offerte su inutilizzateAmazon Elastic Compute Cloud(Amazon EC2) capacità t fino al 90% rispetto al prezzo dell'istanza on-demand ed eseguire le istanze acquisite fino a quando l'offerta supera la correnteprezzo Spot.Amazon EC2cambia periodicamente il prezzo Spot in funzione della domanda e dell'offerta e i clienti che presentano offerte pari o superiori hanno diritto di accedere 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 significativamente il tuoAmazon EC2costi per l'elaborazione in batch, la ricerca scientifica, l'elaborazione delle immagini, la codifica video, la crawling 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:

- Spesso il prezzo che pagherai per ora sarà inferiore alla tua offerta. Amazon EC2 adegua il prezzo Spot periodicamente in base al variare delle richieste in entrata e alla fornitura 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 esegui le istanze Spot e la tua offerta non corrisponde più all'attuale prezzo Spot o lo supera, le istanze saranno terminate. Ciò significa che è necessario assicurarsi che i carichi di lavoro e le applicazioni siano sufficientemente flessibili da sfruttare questa capacità opportunistica.

Le istanze Spot offrono esattamente le stesse prestazioni di tutte le altre istanze Amazon EC2 durante l'esecuzione e, come tutte le altre istanze Amazon EC2 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 la tua offerta e la tua istanza viene terminata daAmazon EC2, non ti verrà addebitato alcun costo per un'ora parziale di utilizzo.

Questo tutorial mostra come utilizzareAWS SDK for Javaper eseguire le operazioni descritte di seguito.

- Inviare una richiesta Spot
- Stabilire quando la richiesta Spot viene soddisfatta
- Annullare la richiesta Spot
- Terminare le istanze associate

Prerequisiti

Per utilizzare questo tutorial, è necessario disporre dellaAWS SDK for Javainstallato, oltre ad aver soddisfatto i suoi prerequisiti di installazione di base. Consulta .Impostazione dell'interfacciaAWS SDK for Javaper ulteriori informazioni.

Fase 1: Configurazione delle credenziali

Per iniziare a utilizzare questo esempio di codice, è necessario configurareAWSCredenziali . Consulta .ConfigurazioneAWSCredenziali e regione per lo sviluppoper istruzioni su come eseguire questa operazione.

> **Note**
>
> Si consiglia di utilizzare le credenziali di un utente IAM per fornire questi valori. Per ulteriori informazioni, consultaRegistrazione adAWSe Creare un utente IAM.

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

Fase 2: Configurazione di un gruppo di sicurezza

UNgruppo di sicurezzaFunziona come un 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 SSH (Secure Shell) dall'indirizzo IP da cui si sta eseguendo l'applicazione. Per configurare un nuovo gruppo di sicurezza, è necessario includere o eseguire il seguente esempio di codice che imposta il gruppo di sicurezza a livello di programmazione.

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

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

Per creare un nuovo gruppo di sicurezza, è necessario eseguire questa applicazione una sola volta.

È inoltre possibile creare il gruppo di sicurezza tramiteAWS Toolkit for Eclipse. Consulta .Gestione di gruppi di sicurezza daAWS Cost Explorerper ulteriori informazioni.

Fase 3: Invio della richiesta Spot

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

Sono disponibili diversi tipi di istanza tra cui scegliere. Vai aAmazon EC2Tipi di istanza 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. Quando abbiamo scritto questo tutorial, utilizzeremo ami-a9d09ed1, l'AMI Amazon Linux più aggiornata disponibile. L'AMI più

recente potrebbe cambiare nel tempo, ma puoi sempre determinare l'AMI della versione più recente seguendo questi passaggi:

1. Aprire la console Amazon EC2.
2. SelezionaAvvia istanza.
3. La prima finestra mostra le AMI disponibili. L'ID AMI è elencato accanto a ciascun titolo AMI. In alternativa, è possibile utilizzare la`DescribeImages`L'API, ma l'utilizzo di questo comando non rientra nell'ambito di questo tutorial.

Ci sono molti modi per affrontare le offerte per le istanze Spot; per ottenere un'ampia panoramica dei vari approcci, è consigliabile guardare ilOfferta per istanze Spotvideo. Tuttavia, per iniziare, descriveremo tre strategie comuni: offerta per assicurarsi un costo inferiore al prezzo on demand; offerta in base al valore del calcolo risultante; offerta per acquisire capacità di elaborazione il più rapidamente possibile.

- Ridurre i costi sotto i prezziHai un processo di elaborazione in batch la cui esecuzione richiede diverse ore o giorni. Tuttavia, hai una certa flessibilità sui tempi di avvio e di completamento. Vuoi vedere se riesci a completarlo a un costo inferiore rispetto alle istanze on demand. Analizzi la cronologia dei prezzi Spot per i tipi di istanza utilizzando la AWS Management Console o le API di Amazon EC2. 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 alternative 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 desideri pagare per le istanze Spot come percentuale del prezzo dell'istanza on demand e pianificare di combinare più istanze avviate in momenti diversi in un'unica richiesta persistente. Se il prezzo specificato è superiore, l'istanza Spot viene terminata. (Più avanti nel tutorial spiegheremo come automatizzare questa operazione).
- Non pagare più del valore del risultatoHai un processo di elaborazione dei dati da completare. 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 tuo tipo di istanza, scegli un prezzo di offerta per cui il costo del tempo di elaborazione non sia superiore al valore dei risultati del processo. Crea un'offerta persistente e impostane l'esecuzione intermittente quando il prezzo Spot raggiunge o scende sotto la tua offerta.

- Acquisire rapidamente le capacità di elaborazioneTi occorre una capacità aggiuntiva imprevista e a breve termine che non è disponibile con le istanze on demand. Dopo aver analizzato la cronologia dei prezzi Spot per il tuo tipo di istanza, fai un'offerta superiore al valore massimo dello storico dei prezzi per offrire un'alta probabilità che la tua richiesta sia soddisfatta in modo rapido e continuare l'elaborazione finché non viene completata.

Dopo aver scelto il prezzo di offerta, sei pronto a richiedere un'istanza Spot. Ai fini di questo tutorial, offriamo il prezzo on demand (0,03 USD) per aumentare al massimo la possibilità che l'offerta sia soddisfatta. Puoi determinare i tipi di istanze disponibili e i prezzi on demand per le istanze, vai aAmazon EC2Pagina dei prezzi. Mentre un'istanza Spot è in esecuzione, paghi il prezzo Spot in corso di validità durante l'esecuzione delle istanze. I prezzi delle istanze Spot sono stabiliti da Amazon EC2 e regolati in modo graduale in base ai trend a lungo termine di offerta e domanda di capacità delle istanze Spot. Puoi anche specificare l'importo che desideri pagare per un'istanza Spot come percentuale del prezzo dell'istanza on demand. Per richiedere un'istanza Spot, devi semplicemente creare la tua richiesta con i parametri scelti in precedenza. Iniziamo creando un`RequestSpotInstanceRequest`oggetto. L'oggetto della richiesta richiede il numero di istanze da avviare e il prezzo di offerta. Inoltre, è necessario impostare la`LaunchSpecification`per la richiesta, che include il tipo di istanza, l'ID dell'AMI e il gruppo di sicurezza da utilizzare. Una volta compilata la richiesta, si chiama il`requestSpotInstances`sul metodo sul`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 dell'istanza Spot. Sono disponibili altre opzioni per configurare le tue richieste Spot. Per ulteriori informazioni, consulta la paginaTutorial: AvanzatoAmazon EC2Gestione delle richieste Spoto ilRequestSpotInstancesclasse nelAWS SDK for JavaInformazioni di riferimento sull'API.

> **Note**
>
> Ti verrà addebitato il costo delle istanze Spot effettivamente avviate, quindi assicurati di annullare tutte le richieste e di terminare tutte le istanze avviate per ridurre i costi associati.

Fase 4: Verifica dello stato della richiesta Spot

Per poter procedere all'ultima fase, vogliamo creare un codice per attendere che la richiesta Spot raggiunga lo stato «attivo». Per determinare lo stato della nostra richiesta Spot, effettuiamo il sondaggiodescribeSpotInstanceRequestsMetodo per lo stato dell'ID della richiesta Spot da monitorare.

L'ID richiesta creato nel passaggio 2 è incorporato nella risposta al nostro`requestSpotInstances`. Il codice di esempio seguente mostra come raccogliere gli ID della richiesta da`requestSpotInstances`risposta e usarli per popolare 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 il tuo ID richiesta, chiama il`describeSpotInstanceRequests`per determinare lo stato di una richiesta. Quindi esegui il ciclo fino a quando la richiesta non è nello stato «aperto». Si noti che monitoriamo per uno stato non «aperto», piuttosto uno stato di, diciamo, «attivo», perché la richiesta può andare direttamente a «chiusa» se c'è un problema con gli argomenti della richiesta. L'esempio di codice seguente 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 completata o non riuscita con un errore che verrà visualizzato sullo schermo. In entrambi i casi, possiamo procedere al passaggio successivo per ripulire tutte le richieste attive e terminare tutte le istanze in esecuzione.

Fase 5: Eliminazione delle richieste e delle istanze Spot

Infine, dobbiamo ripulire le nostre richieste e istanze. È importante annullare tutte le richieste in sospesoeTerminare tutte le istanze La semplice cancellazione delle richieste non termina le istanze, i cui costi continueranno a pagare. Quando termini le istanze, le richieste Spot in genere vengono annullate. Tuttavia, in alcuni scenari, ad esempio quando utilizzi le offerte persistenti, la terminazione delle istanze non è sufficiente a interrompere la rielaborazione della richiesta. Pertanto, è consigliabile sia annullare le offerte attive che terminare le istanze in esecuzione.

Il codice seguente mostra come annullare le tue 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 eventuali istanze in sospeso, è necessario l'ID di istanza associato alla richiesta che le ha avviate. Il seguente esempio di codice prende il nostro codice originale per il monitoraggio delle istanze e aggiunge un`ArrayList`in cui memorizziamo l'ID di istanza associato al`describeInstance`risposta.

```
// 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);
```

Utilizzo degli ID di istanza, memorizzati nel`ArrayList`, terminare 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());
}
```

Mettere tutto insieme

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

Creiamo anche un`GettingStarted`Appclass, che ha un metodo principale in cui eseguiamo le chiamate di funzioni di alto livello. Nello specifico, inizializziamo il`Requests`oggetto descritto in precedenza. Inviare la richiesta dell'istanza Spot. Quindi aspettiamo che la richiesta Spot raggiunga lo stato «Attivo». Infine, puliamo le richieste e le istanze.

Il codice sorgente completo per questo esempio può essere visualizzato o scaricato all'indirizzoGitHub.

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

Fasi successive

Continua con laTutorial: AvanzatoAmazon EC2Gestione delle richieste Spot.

Tutorial: AvanzatoAmazon EC2Gestione delle richieste Spot

Amazon EC2Le istanze Spot ti consentono di fare offerte su inutilizzateAmazon EC2capacità ed esegui tali istanze per tutto il tempo in cui la tua offerta supera la correnteprezzo Spot.Amazon EC2modifica periodicamente il prezzo spot in base all'offerta e alla domanda. Per ulteriori informazioni sulle istanze Spot, consulta.Istanze SpotnellaAmazon EC2Guida per l'utente per le istanze Linux.

Prerequisiti

Per utilizzare questo tutorial è necessario disporre delAWS SDK for Javainstallato, oltre ad aver soddisfatto i suoi prerequisiti di installazione di base. Consulta .Impostazione dell'interfacciaAWS SDK for Javaper ulteriori informazioni.

Configurazione delle credenziali

Per iniziare a utilizzare questo esempio di codice, devi configurareAWSCredenziali . Consulta .ConfigurazioneAWSCredenziali e regione per lo sviluppoper le istruzioni su come eseguire questa operazione.

> **Note**
>
> Ti consigliamo di utilizzare le credenziali di unIAMutente per fornire questi valori. Per ulteriori informazioni, consultaRegistrazione adAWSe crea unIAMUtente.

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

Configurazione di un gruppo di sicurezza

Un gruppo di sicurezza agisce come un firewall che controlla il traffico consentito all'entrata e all'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 SSH (Secure Shell) dall'indirizzo IP da cui si sta eseguendo l'applicazione. Per configurare un nuovo gruppo di sicurezza, è necessario includere o eseguire il seguente esempio di codice che imposta il gruppo di sicurezza a livello di programmazione.

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

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

(Il codice seguente è lo stesso di quello 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 nella`advanced.CreateSecurityGroupApp.java`codice di esempio. Per creare un nuovo gruppo di sicurezza, è necessario eseguire questa applicazione una sola volta.

> **Note**
>
> È anche possibile creare il gruppo di sicurezza tramiteAWS Toolkit for Eclipse. Consulta .Gestione dei gruppi di sicurezza daAWS Cost ExplorernellaAWS Toolkit for EclipseGuida per l'utente per ulteriori informazioni.

Opzioni di creazione delle richieste di istanza Spot

Come abbiamo spiegato inTutorial:Amazon EC2 Istanze Spot, devi creare la tua richiesta con un tipo di istanza, un'Amazon Machine Image (AMI) e l'offerta massima.

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

(Il codice seguente è lo stesso di quello 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 una tantum

Quando si crea una richiesta Spot, è possibile specificare diversi parametri opzionali. Il primo è se la tua richiesta è solo una tantum o persistente. Per impostazione predefinita, è una richiesta una tantum. Una richiesta una tantum può essere soddisfatta una sola volta e una volta terminate le istanze richieste, la richiesta verrà chiusa. Una richiesta persistente viene considerata per l'evasione ogni volta che non esiste un'istanza Spot in esecuzione per la stessa richiesta. Per specificare il tipo di richiesta, è sufficiente impostare la richiesta Type on the Spot. Questa operazione può essere eseguita 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

È inoltre possibile specificare facoltativamente il periodo di tempo in cui la richiesta rimarrà valida. È possibile specificare sia l'ora di inizio che di fine per questo periodo. Per impostazione predefinita,

una richiesta Spot verrà presa in considerazione per l'evasione dal momento in cui viene creata fino a quando non viene soddisfatta o annullata dall'utente. Tuttavia, è possibile limitare il periodo di validità se necessario. Un esempio di come specificare questo periodo è mostrato nel seguente 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 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);
```

Raggruppare il tuoAmazon EC2Richieste di istanza Spot

Hai la possibilità di raggruppare le richieste di istanza Spot in diversi modi. Vedremo i vantaggi derivanti dall'utilizzo di gruppi di lancio, gruppi di zona di disponibilità e gruppi di collocamento.

Se vuoi assicurarti che le tue istanze Spot siano tutte lanciate e terminate insieme, hai la possibilità di sfruttare 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 in un gruppo di lancio sono già state soddisfatte, non vi è alcuna garanzia che vengano soddisfatte anche nuove istanze avviate con lo stesso gruppo di lancio. Nell'esempio di codice seguente è mostrato un esempio di come impostare un gruppo di lancio.

```
// 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 avviate nella stessa zona di disponibilità e non ti interessa quale, puoi sfruttare i gruppi della zona di disponibilità. Un gruppo di zona di disponibilità è un'etichetta che raggruppa un insieme di istanze nella stessa zona di disponibilità. Tutte le istanze che condividono un gruppo di zona di disponibilità e vengono gestite contemporaneamente inizieranno nella stessa zona di disponibilità. Segue un esempio di come impostare un gruppo di 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 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);
```

È possibile specificare una zona di disponibilità desiderata per le istanze Spot. Nell'esempio di codice seguente viene illustrato 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, puoi specificare ungruppo di collocamentose si utilizzano istanze Spot High Performance Computing (HPC), come istanze di calcolo cluster o istanze GPU cluster. Questi gruppi forniscono

connettività a bassa latenza ed elevata larghezza di banda tra le istanze. Segue un esempio di come impostare un gruppo di posizionamento.

```
// 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 facoltativi. È anche importante rendersi conto che la maggior parte di questi parametri, ad eccezione del fatto che l'offerta sia una tantum o persistente, può ridurre la probabilità di evasione delle offerte. Quindi, è

importante sfruttare queste opzioni solo se ne hai bisogno. Tutti gli esempi di codice precedenti sono combinati in un esempio di codice lungo, che può essere trovato nella`com.amazonaws.codesamples.advanced.InlineGettingStartedCodeSampleApp.javaclass`

Come persistere una partizione radice dopo l'interruzione o la cessazione

Uno dei modi più semplici per gestire l'interruzione delle istanze Spot è garantire che i tuoi dati vengano sottoposti a checkpoint su Amazon Elastic Block Store (Amazon).Amazon EBS) volume su cadenza regolare. Con il checkpoint periodicamente, in caso di interruzione si perderanno solo i dati creati dall'ultimo checkpoint (supponendo che non vengano eseguite altre azioni non idempotenti nel mezzo). Per semplificare questo processo, è possibile configurare la richiesta Spot per assicurarsi che la partizione root non venga eliminata in caso di interruzione o interruzione. Nel seguente esempio abbiamo inserito un nuovo codice che mostra come abilitare questo scenario.

Nel codice aggiunto, creiamo un`BlockDeviceMapping`oggetto e imposta il relativo associatoAmazon Elastic Block Store(Amazon EBS) a unAmazon EBSoggetto per cui abbiamo configuratonotessere eliminato se l'istanza Spot viene terminata. Quindi aggiungiamo questo`BlockDeviceMapping`alla ArrayList di 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 di voler ricollegare questo volume alla tua istanza all'avvio, puoi anche utilizzare le impostazioni di mappatura del dispositivo a blocchi. In alternativa, se hai collegato una partizione non root, puoi specificare AmazonAmazon EBSvolumi che vuoi allegare alla tua istanza Spot dopo la ripresa. Puoi farlo semplicemente specificando un ID snapshot nel tuo`EbsBlockDevice`e nome del dispositivo alternativo nel tuo`BlockDeviceMapping`oggetti. Sfruttando i mapping dei dispositivi a blocchi, può essere più semplice avviare l'istanza.

L'utilizzo della partizione root per controllare i dati critici è un ottimo modo per gestire il potenziale di interruzione delle istanze. Per ulteriori metodi per gestire il potenziale di interruzione, visitare il sitoGestione dell'interruzionevideo.

Come taggare le richieste spot e le istanze

Aggiunta di tag aAmazon EC2Le risorse possono semplificare l'amministrazione dell'infrastruttura cloud. Una forma di metadati, i tag possono essere utilizzati per creare nomi user-friendly, migliorare la ricercabilità e migliorare il coordinamento tra più utenti. È anche possibile utilizzare i tag per automatizzare gli script e le parti dei processi. Per saperne di più sul taggingAmazon EC2risorse, vai aUso dei tagnellaAmazon EC2Guida per l'utente per le istanze Linux.

Tagging delle richieste

Per aggiungere tag alle tue richieste spot, devi taggarledoposono stati richiesti. Il valore restituito da`requestSpotInstances()`ti fornisce unRequestSpotInstancesResultoggetto che è possibile utilizzare per ottenere gli ID di richiesta spot per l'etichettatura:

```
// 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 che hai gli ID, puoi taggare le richieste aggiungendo i loro ID aCreateTagsRequeste richiama alAmazon EC2clientecreateTags()Metodo :

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

Tagging di istanze

Analogamente alle richieste spot, è possibile contrassegnare un'istanza solo una volta creata, il che avverrà una volta soddisfatta la richiesta spot (non è più nelaprirestato).

Puoi controllare lo stato delle richieste richiamando ilAmazon EC2clientedescribeSpotInstanceRequests()con un metodoDescribeSpotInstanceRequestsRequestl'oggetto. Il valore restituitoDescribeSpotInstanceRequestsResult Spotl'oggetto contiene un elenco diSpotInstanceRequestoggetti che è possibile utilizzare per eseguire query sullo stato delle richieste spot e ottenere gli ID di istanza una volta che non sono più presenti nelaprirestato.

Una volta che la richiesta spot non è più aperta, è possibile recuperare il suo ID istanza dalSpotInstanceRequestoggetto chiamando il suogetInstanceId()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 di richieste spot e chiusura delle istanze

Annullare una richiesta Spot

Per annullare una richiesta di istanza Spot, chiama`cancelSpotInstanceRequests`sulAmazon EC2cliente con unCancelSpotInstanceRequestsRequestl'oggetto.

```
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());
}
```

Terminazione delle istanze Spot

È possibile terminare qualsiasi istanza Spot in esecuzione passando i loro ID alAmazon EC2clienteterminateInstances()metodo.

```
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 che abbiamo mostrato in questo tutorial in un'unica classe facile da usare. Creiamo un'istanza di una classe chiamata`Requests`che esegue queste operazioni. Creiamo anche un`GettingStartedApp`class, che ha un metodo principale in cui eseguiamo le chiamate di funzioni di alto livello.

Il codice sorgente completo per questo esempio può essere visualizzato o scaricato all'indirizzoGitHub.

Complimenti! Hai completato il tutorial sulle funzionalità di richiesta avanzate per lo sviluppo del software Spot Instance con ilAWS SDK for Java.

Gestione di istanze Amazon EC2

Creazione di un'istanza

Creazione di un nuovoAmazon EC2istanza chiamando AmazonEC2Client`runInstances`metodo, fornendogli unRunInstancesRequestcontenenteAmazon Machine Image (AMI)da usare e untipo 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();
```

Consulta ilEsempio completo.

Avvio di un'istanza

Per avviare unAmazon EC2istanza, chiama AmazonEC2Client`startInstances`metodo, fornendogli unStartInstancesRequestcontenente 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);
```

Consulta ilEsempio completo.

Arresto di un'istanza

Per arrestare unAmazon EC2istanza, chiama AmazonEC2Client`stopInstances`metodo, fornendogli unStopInstancesRequestcontenente l'ID dell'istanza da arrestare.

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);
```

Consulta ilEsempio completo.

Riavvio di un'istanza

Per riavviare unAmazon EC2istanza, chiama AmazonEC2Client`rebootInstances`metodo, fornendogli unRebootInstancesRequestcontenente 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);
```

Consulta ilEsempio completo.

Descrizione delle istanze

Per elencare le istanze, crea unDescribeInstancesRequeste chiama il client AmazonEC2`describeInstances`Metodo. Restituirà unDescribeInstancesResultoggetto che è possibile utilizzare per elencare ilAmazon EC2istanze per l'account e la regione.

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

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 paginati; puoi ottenere altri risultati passando il valore restituito dall'oggetto risultante`getNextToken`metodo per il tuo oggetto richiesta originale`setNextToken`metodo, quindi utilizzando lo stesso oggetto richiesta nella prossima chiamata a`describeInstances`.

Consulta ilEsempio completo.

Monitoraggio di un'istanza

Puoi monitorare diversi aspetti delle istanze Amazon EC2, ad esempio utilizzo della CPU e della rete, memoria disponibile e spazio su disco rimanente. Per ulteriori informazioni sul monitoraggio delle istanze, consultaMonitoraggioAmazon EC2nellaAmazon EC2Guida per l'utente di istanze Linux.

Per iniziare a monitorare un'istanza, è necessario creare unMonitorInstancesRequestcon l'ID dell'istanza da monitorare e passarlo a quello di AmazonEC2Client`monitorInstances`Metodo.

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);
```

Consulta ilEsempio completo.

Arresto del monitoraggio delle istanze

Per interrompere il monitoraggio di un'istanza, creare unUnmonitorInstancesRequestcon l'ID dell'istanza di cui interrompere il monitoraggio e passarlo a AmazonEC2Clientun`monitorInstances`Metodo.

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);
```

Consulta ilEsempio completo.

Ulteriori informazioni

- RunInstancesnellaAmazon EC2Documentazione di riferimento API
- DescribeInstancesnellaAmazon EC2Documentazione di riferimento API
- StartInstancesnellaAmazon EC2Documentazione di riferimento API
- StopInstancesnellaAmazon EC2Documentazione di riferimento API
- RebootInstancesnellaAmazon EC2Documentazione di riferimento API
- MonitorInstancesnellaAmazon EC2Documentazione di riferimento API
- UnmonitorInstancesnellaAmazon EC2Documentazione di riferimento API

Utilizzo di indirizzi IP elastici in Amazon EC2

EC2-Classic sta per andare in pensione

> **Warning**
>
> Ritireremo EC2-Classic il 15 agosto 2022. Suggeriamo di effettuare la migrazione da EC2-Classic a un VPC. Per ulteriori informazioni, consulta Migrare da EC2-Classic a un VPC nella Amazon EC2 User Guide o nellaAmazon EC2 User Guide. Consulta anche il post del 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 di AmazonEC2Client con un AllocateAddressRequestoggetto contenente il tipo di rete (EC2 classico o VPC).

Il valore 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 al metodo di AmazonEC2Client. AssociateAddressRequest`associateAddress`

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 metodo di AmazonEC2Client. `describeAddresses` 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 di AmazonEC2Client, 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, rilasciando un indirizzo IP elastico questo viene automaticamente disassociato da qualsiasi istanza cui è associato. Per dissociare un indirizzo IP elastico senza rilasciarlo, utilizza il metodo di AmazonEC2Client. `disassociateAddress`

Se utilizzi un VPC non di default, devi utilizzare `disassociateAddress` per disassociare l'indirizzo IP elastico prima di provare a rilasciarlo. In caso contrario, Amazon EC2 restituisce un errore (InvalidipAddress). 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

Utilizzo di regioni e zone di disponibilità

Descrivere le regioni

Per visualizzare un elenco delle regioni disponibili per l'account, chiama AmazonEC2Client`describeRegions`Metodo. Restituisce unDescribeRegionsResult. 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());
}
```

Consulta l'elencoEsempio completo.

Descrivere le zone di disponibilità

Per visualizzare un elenco delle zone di disponibilità disponibili per l'account, chiama AmazonEC2Client`describeAvailabilityZones`Metodo. Restituisce unDescribeAvailabilityZonesResult. Chiama il relativo metodo `getAvailabilityZones` per ottenere un elenco di oggetti AvailabilityZone che rappresentano ciascuna 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());
}
```

Consulta l'elencoEsempio completo.

Descrivere gli account

Per descrivere l'account, chiamare il client di AmazonEC2Client`describeAccountAttributes`Metodo. Questo metodo restituisce unDescribeAccountAttributesResultoggetto. Invocare questo metodo `getAccountAttributes` oggetti per ottenere un elenco di oggetti AccountAttribute. È possibile scorrere l'elenco per recuperare un oggetto AccountAttribute.

Puoi ottenere i valori degli attributi del tuo account richiamando ilAccountAttributeoggetto`getAttributeValues`Metodo. Questo metodo restituisce un elenco di oggetti AccountAttributeValue. 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 disponibilitànellaAmazon EC2Guida per l'utente per istanze Linux
- DescribeRegions.nellaAmazon EC2Documentazione di riferimento API
- DescribeAvailabilityZonesnellaAmazon EC2Documentazione di riferimento API

Utilizzo di coppie di chiavi Amazon EC2

Creazione di una coppia di chiavi

Per creare una key pair, chiama il client di AmazonEC2Client`createKeyPair`metodo con unCreateKeyPairRequestche 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);
```

Consulta ilEsempio completo.

Descrizione delle coppie di chiavi

Per visualizzare un elenco delle coppie di chiavi o ottenere le informazioni relative, chiama il client di AmazonEC2Client`describeKeyPairs`metodo. Restituisce unDescribeKeyPairsResultche puoi utilizzare per accedere all'elenco di coppie di chiavi chiamando il`getKeyPairs`metodo, che restituisce un elenco diKeyPairInfooggetti.

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());
}
```

Consulta ilEsempio completo.

Eliminazione di una coppia di chiavi

Per eliminare una key pair, chiama AmazonEC2Client`deleteKeyPair`metodo, passandolo aDeleteKeyPairRequestcontenente il nome della key pair 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);
```

Consulta ilEsempio completo.

Ulteriori informazioni

- Amazon EC2Coppie di chavinellaAmazon EC2Guida per l'utente per istanze Linux
- CreateKeyPairnellaAmazon EC2Documentazione di riferimento API

- DescribeKeyPairsnellaAmazon EC2Documentazione di riferimento API
- DeleteKeyPairnellaAmazon EC2Documentazione di riferimento API

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 di AmazonEC2Client 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 è in grado di controllare il traffico in entrata (ingress) e in uscita (egress) verso le istanze Amazon EC2 dell'utente.

Per aggiungere regole di ingresso al tuo gruppo di sicurezza, utilizza il`authorizeSecurityGroupIngress` metodo di AmazonEC2Client, 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 AuthorizeSecurityGroupEgressRequestin un`authorizeSecurityGroupEgress` metodo AmazonEC2Client.

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 AmazonEC2Client. Restituisce un DescribeSecurityGroupsResultche è possibile utilizzare 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];
```

Guarda l'esempio completo.

Eliminazione di un gruppo di sicurezza

Per eliminare un gruppo di sicurezza, chiama il`deleteSecurityGroup` metodo AmazonEC2Client, passandogli un messaggio DeleteSecurityGroupRequestcontenente 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 EC2Gruppi di sicurezza nella GuidaAmazon EC2 utente per le istanze Linux
- Autorizzazione del traffico in entrata per le istanze Linux nella GuidaAmazon EC2 utente per le istanze Linux
- CreateSecurityGroupnel riferimentoAmazon EC2 API
- DescribeSecurityGroupsnel riferimentoAmazon EC2 API
- DeleteSecurityGroupnel riferimentoAmazon EC2 API
- AuthorizeSecurityGroupIngressnel riferimentoAmazon EC2 API

Esempi di IAMAWS SDK for Java

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

AWS Identity and Access Management(IAM) consente di controllare in modo sicuro l'accessoAWSservizi e risorse per i tuoi utenti. Utilizzando IAM puoi creare e gestireAWSutenti e gruppi e utilizza le autorizzazioni per consentire e negare l'accessoAWSrisorse AWS. Per una guida completa a IAM, consulta laIAMGuida per l'utente di.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ciascuna 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 AmazonIdentityManagementClient`createAccessKey`con`CreateAccessKeyRequest`oggetto.

`CreateAccessKeyRequest`ha due costruttori, uno che prende un nome utente e un altro senza parametri. Se si utilizza la versione che non richiede parametri, è necessario impostare il nome utente utilizzando il`withUserName`setter method (metodo 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, crea unListAccessKeysRequestoggetto contenente il nome utente per cui elencare le chiavi e passarlo a AmazonIdentityManagementClient`listAccessKeys`metodo.

> **Note**
>
> Se non si fornisce un nome utente `alistAccessKeys`, tenterà di elencare le chiavi di accesso associate alAccount AWSche 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). Puoi chiamare`getIsTruncated`sul valore restituitoListAccessKeysResultper vedere se la query ha restituito meno risultati di quelli disponibili. Se è così, chiama`setMarker`sul`ListAccessKeysRequest`e riportalo alla prossima invocazione di`listAccessKeys`.

Vedi l'esempio completo su GitHub.

Recupero dell'ora ultimo utilizzo di una chiave di accesso

Per ottenere l'ora dell'ultimo utilizzo di una chiave di accesso, chiama AmazonIdentityManagement`Client`getAccessKeyLastUsed`con l'ID della chiave di accesso (che può essere passato utilizzando unGetAccessKeyLastUsedRequestoggetto o direttamente al sovraccarico che prende direttamente l'ID della chiave di accesso.

È quindi possibile utilizzare il prodotto restituitoGetAccessKeyLastUsedResultper recuperare l'ora ultimo utilizzo 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 unUpdateAccessKeyRequestobject, che fornisce l'ID chiave di accesso, facoltativamente il nome utente e il desideratoStato, quindi passando l'oggetto della richiesta a AmazonIdentityManagementClient`updateAccessKey`metodo.

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 AmazonIdentityManagementClient`deleteKey`metodo, fornendogli unDeleteAccessKeyRequestcontenente 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 nuovamente attivata in seguito, utilizza invece il metodo updateAccessKey.

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 documento di riferimento delle API IAM
- ListAccessKeys,nel documento di riferimento delle API IAM
- GetAccessKeyLastUsednel documento di riferimento delle API IAM
- UpdateAccessKeynel documento di riferimento delle API IAM
- DeleteAccessKeynel documento di riferimento delle API IAM

Gestione degli utenti IAM

Creazione di un utente

Crea un nuovo utente IAM fornendo il nome utente al`createUser` metodoAmazonIdentityManagementClient'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 suGitHub.

Elencazione di utenti

Per elencare gli utenti IAM del tuo account, creane uno nuovo ListUsersRequeste passalo alAmazonIdentityManagementClient`listUsers` metodo. È possibile recuperare l'elenco degli utenti`getUsers` richiamando 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 ritorna`true`, chiama il`setMarker()` metodo dell'oggetto della 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 suGitHub.

Aggiornamento di un utente

Per aggiornare un utente, chiama il`updateUser` metodo dell'AmazonIdentityManagementClientoggetto, che accetta un UpdateUserRequestoggetto che puoi 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 suGitHub.

Eliminazione di un utente

Per eliminare un utente,AmazonIdentityManagementClient chiama la`deleteUser` richiesta con un set di UpdateUserRequestoggetti 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 suGitHub.

Ulteriori informazioni

- Utenti IAM nella Guida per l'IAMutente
- Gestione degli utenti IAM nella Guida per l'IAMutente
- CreateUsernel 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 per la pagina di accesso contenga il nome della tua azienda o un altro identificatore descrittivo invece dell'ID dell', puoi creare un alias per la pagina di accesso contenga il nome della tua azienda o un altro identificatore descrittivo invece dell'Account AWSID dell', puoi creare un alias per la tuaAccount AWS azienda

> **Note**
>
> AWSsupporta esattamente un alias per account.

Creazione di un alias dell'account

Per creare un alias di account, chiama il`createAccountAlias` metodoAmazonIdentityManagementClient'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 suGitHub.

Elencazione di alias dell'account

Per elencare alias dell'account, se presente, chiama il metodo `listAccountAliases` di AmazonIdentityManagementClient.

> **Note**
>
> Il restituito ListAccountAliasesResultsupporta gli stessi`getMarker` metodi`getIsTruncated` andAWS SDK for Java degli altri metodi di elenco, maAccount 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 suGitHub.

Eliminazione di un alias dell'account

Per eliminare l'alias dell'account, chiama il metodo `deleteAccountAlias` di AmazonIdentityManagementClient. Quando si elimina un alias di account, è necessario fornirne il nome utilizzando un DeleteAccountAliasRequestoggetto.

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 suGitHub.

Ulteriori informazioni

- L'IDAWS dell'account e il relativo alias nella Guida per l'IAMutente
- 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 inCreatePolicyRequestal client AmazonIdentityManagement`createPolicy`metodo.

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 delle policy IAM sono stringhe JSON con unsintassi 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 politica esistente, chiama AmazonIdentityManagementClient`getPolicy`, fornendo l'ARN della policy all'interno diGetPolicyRequestoggetto.

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 un IAM (http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html[role] chiamando AmazonIdentityManagementClient`attachRolePolicy`, fornendo il nome del ruolo e l'ARN della policy in unAttachRolePolicyRequest allegata.

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

Puoi elencare policy collegate su un ruolo chiamando AmazonIdentityManagementClient`listAttachedRolePolicies`metodo. Questo accetta un oggetto ListAttachedRolePoliciesRequest contenente il nome del ruolo per il quale elencare le policy.

Esegui una chiamata a`getAttachedPolicies`sul valore restituitoRisultati ListAttachedRolePoliciesResultPer ottenere l'elenco delle policy collegate. I risultati possono essere troncati; se il`ListAttachedRolePoliciesResult`oggetto`getIsTruncated`restituisce il metodo`true`, chiama il`ListAttachedRolePoliciesRequest`oggetto`setMarker`metodo e usalo per chiamare`listAttachedRolePolicies`di nuovo per ottenere il batch successivo 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 distaccare una policy da un ruolo, chiama AmazonIdentityManagementClient`detachRolePolicy`, fornendo il nome del ruolo e l'ARN della policy in unDetachRolePolicyRequest.

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 policy IAMnellaIAMGuida per l'utente di .
- AWSRiferimento alla politica IAMnellaIAMGuida per l'utente di .
- CreatePolicynel documento di riferimento dell'API IAM
- GetPolicynel documento di riferimento dell'API IAM
- AttachRolePolicynel documento di riferimento dell'API IAM
- ListAttachedRolePoliciesnel documento di riferimento dell'API IAM
- DetachRolePolicynel documento di riferimento dell'API IAM

Utilizzo dei certificati del server IAM

Per abilitare connessioni HTTPS al sito Web o all'applicazione suAWS, hai bisogno di un oggetto SSL/TLSCertificato del server. Puoi utilizzare un certificato del server fornito daAWSCertificate Manager o uno ottenuto da un provider esterno.

Ti consigliamo di utilizzare ACM per assegnare, gestire e distribuire certificati del server. Con ACM puoi richiedere un certificato, distribuirlo nellaAWSrisorse e lascia che ACM gestisca automaticamente i rinnovi dei certificati. I certificati forniti da ACM sono gratuiti. Per ulteriori informazioni su ACM, consulta laGuida per l'utente di ACM.

Ottenere un certificato del server

Puoi recuperare un certificato del server chiamandogetServerCertificatemetodo, passandolo aGetServerCertificateRequestcon 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 pubblicare certificati del server, chiama AmazonIdentityManagementClient`listServerCertificates`metodo conListServerCertificatesRequest. Restituisce unListServerCertificatesResult.

Chiamare il metodo `getServerCertificateMetadataList` dell'oggetto `ListServerCertificateResult` restituito per ottenere un elenco di oggetti ServerCertificateMetadata che puoi utilizzare per ottenere informazioni su ogni certificato.

I risultati possono essere troncati; se`ListServerCertificateResult`oggetto`getIsTruncated`restituisce il metodo`true`, chiama`ListServerCertificatesRequest`oggetto`setMarker`metodo e usalo per chiamare`listServerCertificates`di nuovo per ottenere il batch successivo 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

Puoi aggiornare il nome o il percorso di un certificato del server chiamando AmazonIdentityManagementClient`updateServerCertificate`metodo. Questo accetta un oggetto UpdateServerCertificateRequest 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 AmazonIdentityManagementClient`deleteServerCertificate`metodo conDeleteServerCertificateRequestcontenente 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 del servernellaIAMGuida per l'utente di
- GetServerCertificatenella Documentazione di riferimento delle API di
- ListServerCertificatesnella Documentazione di riferimento delle API di
- UpdateServerCertificatenella Documentazione di riferimento delle API di
- DeleteServerCertificatenella Documentazione di riferimento delle API di
- Guida per l'utente di ACM

Esempi di Lambda utilizzando l'AWS SDK for Java

In questa sezione vengono forniti esempi di programmazione di Lambda utilizzando AWS SDK for Java.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ciascuna 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

Richiamo, elencare ed eliminareLambdaFunzioni

In questa sezione vengono forniti esempi di programmazione conLambdaclient di servizio utilizzando ilAWS SDK for Java. Per scoprire come creare unLambdaconsulta la funzioneCome creareAWS Lambdafunzioni.

Argomenti

Richiamo di una funzione

È possibile invocare unLambdafunzione creando unAWSLambdaobiettare e invocare il suo`invoke`metodo. Creare un oggetto InvokeRequest per specificare informazioni aggiuntive, ad esempio il nome della funzione e il payload da passare alla funzione Lambda. I nomi delle funzioni vengono visualizzati comearn:aws:lambda:us-east- 1:55556330391:function:HelloFunction. È possibile recuperare il valore osservando la funzione nellaAWS Management Console.

Per passare i dati del payload a una funzione, invocareInvokeRequest richiamodell'oggetto`withPayload`Metodo e specificare una stringa in formato JSON, come mostrato nell'esempio di codice seguente.

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

Nell'esempio di codice riportato di seguito viene illustrato come invocare 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 unAWSLambdaobiettare e invocare il suolistFunctionsmetodo. Questo metodo restituisce unListFunctionsResultoggetto. È possibile richiamare il metodo `getFunctions` di questo oggetto per restituire un elenco di oggetti FunctionConfiguration. 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 codice di esempio Java illustra come recuperare un elenco diLambdanomi delle 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 unAWSLambdaobiettare e invocare il suodeleteFunctionmetodo. Creare un oggetto DeleteFunctionRequest e passarlo al metodo `deleteFunction`. Questo oggetto contiene informazioni quali il nome della funzione da eliminare. I nomi delle funzioni vengono visualizzati comearn:aws:lambda:us-east- 1:55556330391:function:HelloFunction. È possibile recuperare il valore osservando la funzione nellaAWS Management Console.

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 illustra come eliminare unLambdafunzione.

```
        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.

Esempi di Amazon Pinpoint utilizzando l'AWS SDK for Java

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

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ciascuna 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 inAmazon Pinpoint

Un'app è unaAmazon Pinpointprogetto in cui definisci il pubblico per un'applicazione distinta e coinvolgi questo pubblico con messaggi personalizzati. Gli esempi in questa pagina dimostrano come creare una nuova app o eliminarne una esistente.

Creazione di un'app

Crea una nuova app inAmazon Pinpointfornendo il nome di un'app alCreateAppRequestoggetto e quindi passando quell'oggetto a quello di AmazonPinpointClient`createApp`metodo.

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, chiama AmazonPinpointClient`deleteApp`una richiesta con unDeleteAppRequestoggetto 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);
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- AppnellaAmazon PinpointDocumentazione di riferimento API

- AppnellaAmazon PinpointDocumentazione di riferimento API

Creazione di endpoint inAmazon Pinpoint

Un endpoint identifica in modo univoco un dispositivo dell'utente al quale puoi inviare notifiche push conAmazon Pinpoint. Se la tua app è abilitata conAmazon Pinpointsupporto, la tua app registra automaticamente un endpoint conAmazon Pinpointquando un nuovo utente apre l'app. L'esempio seguente illustra come aggiungere un nuovo endpoint a livello di programmazione.

Creare un endpoint

Creare un nuovo endpoint inAmazon Pinpointfornendo i dati endpoint in unEndpointRequestl'oggetto.

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 unUpdateEndpointRequestoggetto con quello EndpointRequest l'oggetto. Infine, passa il UpdateEndpointRequest opporsi a AmazonPinpointClient`updateEndpoint`metodo.

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

- Aggiunta di endpointnellaAmazon PinpointGuida per gli sviluppatori
- Punto finalenellaAmazon PinpointDocumentazione di riferimento API

Creazione di segmenti inAmazon Pinpoint

Un segmento di utenti rappresenta un sottoinsieme degli utenti basato su caratteristiche condivise, ad esempio l'ultimo volta in cui ha aperto l'app o il dispositivo utilizzato. L'esempio seguente dimostra come definire un segmento di utenti.

Creazione di un segmento

Crea un nuovo segmento inAmazon Pinpointdefinendo le dimensioni del segmento in aSegmentDimensionsoggetto.

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);
```

Successivo imposta ilSegmentDimensionsoggetto inWriteSegmentRequest, che a sua volta viene utilizzato per creare unCreateSegmentRequestoggetto. Quindi passa l' CreateSegmentRequest opporsi a AmazonPinpointClient`createSegment`metodo.

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 PinpointSegmentinellaAmazon PinpointGuida per l'utente di
- Creazione di segmentinellaAmazon PinpointGuida per gli sviluppatori
- SegmentinellaAmazon PinpointDocumentazione di riferimento API

- SegmentnellaAmazon PinpointDocumentazione di riferimento API

Creazione di campagne inAmazon Pinpoint

Puoi utilizzare le campagne per aumentare il coinvolgimento tra la tua app e i tuoi utenti. Puoi creare una campagna per raggiungere un particolare segmento dei tuoi utenti con messaggi personalizzati o promozioni speciali. Questo esempio dimostra come creare una nuova campagna standard che invia una notifica push personalizzata a un segmento specificato.

Creazione di una campagna

Prima di creare una nuova campagna, è necessario 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 campagnaAmazon Pinpoint WriteCampaignRequestfornendo la configurazione della campagna a un CreateCampaignRequestoggetto. Infine, passa l'CreateCampaignRequestoggettoAmazonPinpointClient al`createCampaign` metodo.

Codice

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

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

Vedi l'esempio completo suGitHub.

Ulteriori informazioni

- Amazon PinpointCampagne nella Guida per l'Amazon Pinpointutente
- Creazione di campagne nella Guida perAmazon Pinpoint gli sviluppatori
- Campagne nell'Amazon PinpointAPI Reference
- Campagna nell'Amazon PinpointAPI Reference
- Attività della campagna nell'Amazon PinpointAPI Reference
- Versioni delle campagne nel riferimentoAmazon Pinpoint API
- Versione della campagna nel riferimentoAmazon Pinpoint API

Aggiornamento dei canali inAmazon Pinpoint

Un canale definisce i tipi di piattaforme a cui è possibile recapitare i messaggi. L'esempio mostra come utilizzare il canale APN per inviare un messaggio.

Aggiornamento di un canale

Abilita un canale inAmazon Pinpointfornendo un ID app e un oggetto richiesta del tipo di canale che si desidera aggiornare. Questo esempio aggiorna il canale APN, che richiede ilRichiesta

canale APNSoggetto. Imposta queste nelUpdateApnsChannelRequeste passa quell'oggetto a AmazonPinpointClient`updateApnsChannel`metodo.

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 PinpointCanalinellaAmazon PinpointGuida per l’utente di
- Canale ADMnellaAmazon PinpointDocumentazione di riferimento API
- Canale APNnellaAmazon PinpointDocumentazione di riferimento API
- Canale Sandbox APNnellaAmazon PinpointDocumentazione di riferimento API
- Canale VoIP APNnellaAmazon PinpointDocumentazione di riferimento API
- Canale Sandbox VoIP APNnellaAmazon PinpointDocumentazione di riferimento API
- BaidunellaAmazon PinpointDocumentazione di riferimento API
- Canale e-mail dinellaAmazon PinpointDocumentazione di riferimento API
- Canale GCMnellaAmazon PinpointDocumentazione di riferimento API
- Canale SMS dinellaAmazon PinpointDocumentazione di riferimento API

Esempi di Amazon S3 utilizzando l'AWS SDK for Java

In questa sezione vengono forniti esempi di programmazione di Amazon S3 utilizzando AWS SDK for Java.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ciascuna 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, elencazione ed eliminazioneAmazon S3Bucket

Ogni oggetto (file) inAmazon S3deve risiedere all'interno di unsecchio, che rappresenta una raccolta (container) di oggetti. Ogni secchio è conosciuto dachiave(name), deve essere univoco. Per ulteriori informazioni sui bucket e la relativa configurazione, consultaUtilizzo diAmazon S3BucketnellaAmazon Simple Storage ServiceGuida per l'utente di .

> **Note**
>
> Best practice
> Ti consigliamo di abilitare la regola del ciclo di vita AbortIncompleteMultipartUpload sui bucket Amazon S3.
> Questa regola indica a Amazon S3 di interrompere l'esecuzione di caricamenti in più parti che non sono stati completati entro un determinato numero di giorni dopo l'avvio. Quando questo

limite di tempo impostato viene superato, Amazon S3 interrompe il caricamento ed elimina i dati di caricamento incompleti.
Per ulteriori informazioni, consultaConfigurazione del ciclo di vita per un bucket con funzione Versioninella Amazon S3Guida per l'utente di .

> **Note**
>
> Questi esempi di codice presuppongono che tu comprenda il materiale inUtilizzo diAWS SDK for Javae hanno configurato il valore predefinitoAWSCredenziali che utilizzano le informazioni inConfigurazioneAWSCredenziali e regione per lo sviluppo.

Creare un bucket

Usa il client Amazon S3`createBucket`Metodo. Il nuovoBucketrestituisce. L`createBucket`il 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, chiamare il`doesBucketExist`Metodo. Restituirà`true`se il secchio esiste, e`false`In caso contrario, .

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

Usa il client Amazon S3 `listBuckets` Metodo. In caso positivo, un elenco di `Bucket` restituisce.

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 eliminare un Amazon S3 bucket, occorre accertarsi che il bucket sia vuoto o che si verifichi un errore. Se disponi di un secchio versione, occorre eliminare anche gli eventuali oggetti con versione associati al bucket.

> **Note**
>
> LaEsempio completoinclude ciascuno di questi passaggi in ordine, fornendo una soluzione completa per eliminare unAmazon S3secchio e il suo contenuto.

Argomenti

- Rimuovere oggetti da un bucket senza versione prima di eliminarlo
- Rimuovere gli oggetti da un bucket con versione prima di eliminarlo
- Eliminazione di un bucket vuoto

Rimuovere oggetti da un bucket senza versione prima di eliminarlo

Usa il client Amazon`S3`Metodo `listObjects` per recuperare l'elenco di oggetti e`deleteObject`per eliminarne ciascuno.

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;
    }
}
```

Vedi l'esempio completo su GitHub.

Rimuovere gli oggetti da un bucket con versione prima di eliminarlo

Se si sta utilizzando unsecchio versione, è inoltre necessario rimuovere qualsiasi versione memorizzata degli oggetti nel bucket prima che il bucket possa essere eliminato.

Utilizzando uno schema simile a quello utilizzato per la rimozione di oggetti all'interno di un bucket, rimuovi gli oggetti con versione utilizzando il client Amazon`S3``listVersions`metodo per elencare tutti gli oggetti con versione e quindi`deleteVersion`per eliminarne ciascuno.

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;
    }
}
```

Vedi l'esempio completo su GitHub.

Eliminazione di un bucket vuoto

Dopo aver rimosso gli oggetti da un bucket (inclusi gli oggetti con versione), puoi eliminare il bucket stesso utilizzando il client AmazonS3`deleteBucket`Metodo.

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);
```

Vedi 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 secchio.

> **Note**
>
> Questi esempi di codice presuppongono che l'utente abbia compreso il materiale contenuto in Utilizzo delle credenziali predefinite AWS SDK for Java e abbia configurato AWS le credenziali predefinite utilizzando le informazioni contenute in Configurazione AWS delle credenziali e Regione per lo sviluppo.

Argomenti

Caricamento di un oggetto

Usa il `putObject` metodo del client AmazonS3, fornendo il nome del bucket, il nome della chiave e il file da caricare. Il bucket deve esistere; in caso contrario si verifica 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);
}
```

Guarda l'esempio completo suGitHub.

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());
}
```

Guarda l'esempio completo suGitHub.

Donwload di un oggetto

Usa il `getObject` metodo del client AmazonS3, passandogli il nome di un bucket e di un oggetto da scaricare. In caso di successo, il metodo restituisce un S3Object. Il bucket e la chiave dell'oggetto specificati devono esistere; in caso contrario si verifica un errore.

È possibile ottenere il contenuto dell'oggetto `getObjectContent` richiamando il`S3Object`. Questo 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);
}
```

Guarda l'esempio completo suGitHub.

Copiare, spostare o rinominare oggetti

Puoi copiare un oggetto da un bucket all'altro. `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!");
```

Guarda l'esempio completo suGitHub.

> **Note**
>
> È possibile utilizzarlo `copyObject` con DeleteObject per spostare o rinominare un oggetto, copiando prima l'oggetto con un nuovo nome (è possibile utilizzare lo stesso bucket sia come origine che come destinazione) e quindi eliminando l'oggetto dalla sua vecchia posizione.

Eliminazione di un oggetto

Usa il `deleteObject` metodo del client AmazonS3, passandogli il nome di un bucket e di un oggetto da eliminare. Il bucket e la chiave dell'oggetto specificati devono esistere; in caso contrario si verifica 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);
}
```

Guarda l'esempio completo suGitHub.

Elimina più oggetti contemporaneamente

Utilizzando il metodo del client AmazonS3, puoi eliminare più oggetti dallo stesso bucket passando i loro nomi al link: `deleteObjects` sdk-for-java metodo /v1/reference/com/amazonaws/services/s3/model/ .html. 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 suGitHub.

GestioneAmazon S3Autorizzazioni di accesso per bucket e oggetti

È possibile utilizzare le liste di controllo accessi (ACL) perAmazon S3benne e oggetti per un controllo a grana fine sul tuoAmazon S3risorse AWS.

> **Note**
>
> Questi esempi di codice presuppongono che tu comprenda il materiale inUtilizzo diAWS SDK for Javae hanno configurato il valore predefinitoAWScredenziali che utilizzano le informazioni inConfigurazioneAWSCredenziali e regione per lo sviluppo.

Ottieni l'elenco di controllo degli accessi per un bucket

Per ottenere l'ACL corrente per un bucket, chiama AmazonS3`getBucketAcl`metodo, passandolo ilnome bucketper interrogare. Questo metodo restituisce unAccessControlListoggetto. Per ottenere ogni concessione di accesso nell'elenco, chiamatela`getGrantsAsList`metodo, che restituirà un elenco Java standard diGrantoggetti.

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.

Impostazione dell'elenco di controllo degli accessi per un bucket

Per aggiungere o modificare le autorizzazioni a un ACL per un bucket, chiama AmazonS3`setBucketAcl`metodo. Ci vuole unAccessControlListoggetto che contiene un elenco di beneficiari 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**
>
> È possibile fornire l'identificatore univoco del beneficiario direttamente utilizzando ilGranteeclasse o usa ilEmailAddressGranteeclasse per impostare il beneficiario via e-mail, come abbiamo fatto qui.

Vedi l'esempio completo su GitHub.

Ottieni l'elenco di controllo degli accessi per un oggetto

Per ottenere l'ACL corrente per un oggetto, chiama AmazonS3`getObjectAcl`metodo, passandolo ilnome bucketeil nome dell'oggettoper interrogare. LIKE`getBucketAcl`, questo metodo restituisce unAccessControlListoggetto che è possibile utilizzare per esaminare ciascunoGrant.

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.

Impostazione dell'elenco di controllo di accesso per un oggetto

Per aggiungere o modificare le autorizzazioni a un ACL per un oggetto, chiama AmazonS3`setObjectAcl`metodo. Ci vuole unAccessControlListoggetto che contiene un elenco di beneficiari 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**
>
> È possibile fornire l'identificatore univoco del beneficiario direttamente utilizzando ilGranteeclasse o usa ilEmailAddressGranteeclasse per impostare il beneficiario via e-mail, come abbiamo fatto qui.

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- GET Bucket aclnellaAmazon S3Documentazione di riferimento API
- PUT Bucket aclnellaAmazon S3Documentazione di riferimento API
- GET Object aclnellaAmazon S3Documentazione di riferimento API
- PUT Object aclnellaAmazon S3Documentazione di riferimento API

Gestione degli accessi aAmazon S3Bucket utilizzando policy di bucket

È possibile impostare, ottenere o eliminare unpolicy di bucketper gestire l'accesso al tuoAmazon S3Bucket.

Impostare una policy di bucket

È possibile impostare una policy di bucket per un determinato bucket S3 tramite:

- Chiamare il client AmazonS3`setBucketPolicy`e fornendogli unSetBucketPolicyRequest
- Impostazione diretta del criterio utilizzando il`setBucketPolicy`overload che accetta il nome del bucket e il testo del criterio (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);
}
```

Utilizzare la classe di criteri per generare o convalidare un criterio

Quando fornisce una policy di bucket a`setBucketPolicy`è possibile effettuare le seguenti operazioni:

- Specificare il criterio direttamente come stringa di testo formattato JSON
- Crea il criterio utilizzando ilPolicyclasse

Usando il`Policy`class, non devi preoccuparti della formattazione corretta della stringa di testo. Per ottenere il testo della policy JSON dal`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`fornisce inoltre una`fromJson`metodo che può tentare di creare un criterio utilizzando una stringa JSON passata. Il metodo lo convalida per garantire che il testo possa essere trasformato in una struttura di criteri valida e fallirà con un`IllegalArgumentException`se il testo del criterio 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 un criterio letto da un file o da altri mezzi.

Vedi l'esempio completo su GitHub.

Ottenere una policy di bucket

Per recuperare una policy per unAmazon S3bucket, chiama il client AmazonS3`getBucketPolicy`metodo, passando il nome del bucket da cui ottenere la polizza.

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 si ha accesso ad esso, o se non ha criteri di bucket,`AmazonServiceException`viene generato.

Vedi l'esempio completo su GitHub.

Eliminare una policy del bucket

Per eliminare una politica del bucket, chiama il client Amazon S3`deleteBucketPolicy`fornisce 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 ha esito positivo anche se il bucket non ha già una politica. Se si specifica un nome di bucket che non esiste o se non si ha accesso al bucket,`AmazonServiceException`viene generato.

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Panoramica della sintassi della policy di accessonellaAmazon Simple Storage ServiceGuida per l'utente di
- Esempi di policy di bucketnellaAmazon Simple Storage ServiceGuida per l'utente di

Utilizzo di TransferManager perAmazon S3Operazioni

Puoi utilizzare il pluginAWS SDK for Java TransferManager classe per trasferire in modo affidabile i file dall'ambiente locale aAmazon S3e per copiare oggetti da una posizione S3 a un'altra.`TransferManager`può ottenere lo stato di avanzamento di un trasferimento e mettere in pausa o riprendere caricamenti e download.

> **Note**
>
> Best practice
> Ti consigliamo di abilitare la regola del ciclo di vita AbortIncompleteMultipartUpload sui bucket Amazon S3.
> Questa regola indica a Amazon S3 di interrompere l'esecuzione di caricamenti in più parti che non sono stati completati entro un determinato numero di giorni dopo l'avvio. Quando questo limite di tempo impostato viene superato, Amazon S3 interrompe il caricamento ed elimina i dati di caricamento incompleti.
> Per ulteriori informazioni, consultaConfigurazione del ciclo di vita per un bucket con funzione Versioni multiplenellaAmazon S3Guida per l'utente di .

> **Note**
>
> Questi esempi di codice presuppongono che tu comprenda il materiale inUtilizzo diAWS SDK for Javae hanno configurato il valore predefinitoAWScredenziali che utilizzano le informazioni inConfigurazioneAWSCredenziali e regione per lo sviluppo.

Caricamento di file e directory

TransferManager può caricare file, elenchi di file e directory su qualsiasiAmazon S3secchi che haicreato in precedenza.

Argomenti

- Caricamento di un file singolo
- Carica un elenco di file
- Caricamento di una directory

Caricamento di un file singolo

Chiama TransferManager`upload`metodo, fornendo unAmazon S3nome bucket, nome chiave (oggetto) e Java standardFileoggetto 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();
```

La`upload`restituisce il metodoimmediatamente, fornendo un`Upload`oggetto da utilizzare per verificare lo stato di trasferimento o per attenderne il completamento.

Consulta .Attendi il completamento di un trasferimentoper informazioni sull'uso di`waitForCompletion`per completare con successo un trasferimento prima di chiamare TransferManager`shutdownNow`metodo. In attesa del completamento del trasferimento, è possibile effettuare il sondaggio o ascoltare aggiornamenti sullo stato e sui progressi. Consulta .Ottieni stato di trasferimento e avanzamentoper ulteriori informazioni.

Vedi l'esempio completo su GitHub.

Carica un elenco di file

Per caricare più file in un'unica operazione, chiama TransferManager`uploadFileList`metodo, fornendo quanto segue:

- Un recordAmazon S3nome bucket
- UNprefisso della chiaveper anticipare i nomi degli oggetti creati (il percorso all'interno del bucket in cui posizionare gli oggetti)
- UNFileoggetto che rappresenta la directory relativa da cui creare percorsi file
- UNElencooggetto contenente un set diFileoggetti 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();
```

Consulta .Attendi il completamento di un trasferimentoper informazioni sull'uso di`waitForCompletion`per completare con successo un trasferimento prima di chiamare TransferManager`shutdownNow`metodo. In attesa del completamento del trasferimento, è possibile effettuare il sondaggio o ascoltare aggiornamenti sullo stato e sui progressi. Consulta .Ottieni stato di trasferimento e avanzamentoper ulteriori informazioni.

LaMultipleFileUploadoggetto restituito da`uploadFileList`può essere utilizzato per interrogare lo stato di trasferimento o lo stato di avanzamento. Consulta .Sondaggio dell'attuale avanzamento di un trasferimentoeOttieni l'avanzamento del trasferimento con un ProgressListenerper ulteriori informazioni.

È possibile utilizzare anche`MultipleFileUpload`'s`getSubTransfers`metodo per ottenere l'individuo`Upload`oggetti per ogni file trasferito. Per ulteriori informazioni, consultaOttieni lo stato di avanzamento dei subtrasferimenti.

Vedi l'esempio completo su GitHub.

Caricamento di una directory

Puoi utilizzare TransferManager`uploadDirectory`metodo per caricare un'intera directory di file, con la possibilità di copiare i file nelle sottodirectory in modo ricorsivo. Fornisci unAmazon S3nome del bucket, key prefix S3, aFileoggetto che rappresenta la directory locale da copiare e a`boolean`valore che indica se si desidera copiare le sottodirectory in modo ricorsivo (veroofalso).

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();
```

Consulta .Attendi il completamento di un trasferimentoper informazioni sull'uso di`waitForCompletion`per completare con successo un trasferimento prima di chiamare TransferManager`shutdownNow`metodo. In attesa del completamento del trasferimento, è possibile effettuare il sondaggio o ascoltare aggiornamenti sullo stato e sui progressi. Consulta .Ottieni stato di trasferimento e avanzamentoper ulteriori informazioni.

LaMultipleFileUploadoggetto restituito da`uploadFileList`può essere utilizzato per interrogare lo stato di trasferimento o lo stato di avanzamento. Consulta .Sondaggio dell'attuale avanzamento di un trasferimentoeOttieni l'avanzamento del trasferimento con un ProgressListenerper ulteriori informazioni.

È possibile utilizzare anche`MultipleFileUpload`'s`getSubTransfers`metodo per ottenere l'individuo`Upload`oggetti per ogni file trasferito. Per ulteriori informazioni, consultaOttieni lo stato di avanzamento dei subtrasferimenti.

Vedi l'esempio completo su GitHub.

Download di file o directory

Utilizzo dell' TransferManager classe per scaricare un singolo file (Amazon S3object) o una directory (unAmazon S3nome del bucket (seguito da un prefisso oggetto) daAmazon S3.

Argomenti

- Download di un file singolo
- Scarica una directory

Download di un file singolo

Usa il TransferManager`download`metodo, fornendo ilAmazon S3nome bucket contenente l'oggetto che si desidera scaricare, il nome della chiave (oggetto) e unFileoggetto 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();
```

Consulta .Attendi il completamento di un trasferimentoper informazioni sull'uso di`waitForCompletion`per completare con successo un trasferimento prima di chiamare TransferManager`shutdownNow`metodo. In attesa del completamento del trasferimento, è possibile effettuare il sondaggio o ascoltare aggiornamenti sullo stato e sui progressi. Consulta .Ottieni stato di trasferimento e avanzamentoper ulteriori informazioni.

Vedi l'esempio completo su GitHub.

Scarica una directory

Per scaricare un set di file che condividono un key prefix comune (analogo a una directory su un file system) daAmazon S3, usa TransferManager`downloadDirectory`metodo. Il metodo accetta ilAmazon S3nome bucket contenente gli oggetti che si desidera scaricare, il prefisso dell'oggetto condiviso da tutti gli oggetti eFileoggetto che rappresenta la directory in cui scaricare i file nel sistema locale. Se la directory con nome 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();
```

Consulta .Attendi il completamento di un trasferimentoper informazioni sull'uso di`waitForCompletion`per completare con successo un trasferimento prima di chiamare TransferManager`shutdownNow`metodo. In attesa del completamento del trasferimento, è possibile effettuare il sondaggio o ascoltare aggiornamenti sullo stato e sui progressi. Consulta .Ottieni stato di trasferimento e avanzamentoper ulteriori informazioni.

Vedi l'esempio completo su GitHub.

Copia oggetti

Per copiare un oggetto da un bucket S3 in un altro, utilizzare 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 di un trasferimento

Se l'applicazione (o il thread) può bloccarsi fino al completamento del trasferimento, è possibile utilizzare iltrasferimentointerfacciawaitForCompletionmetodo per bloccare fino al completamento del trasferimento o si verifica 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);
}
```

Si ottengono progressi nei trasferimenti se effettui un sondaggio per eventiprimavocazionewaitForCompletion, implementare un meccanismo di polling su un thread separato o ricevere gli aggiornamenti dello stato di avanzamento in modo asincrono utilizzando unProgressListener.

Vedi l'esempio completo su GitHub.

Ottieni stato di trasferimento e avanzamento

Ciascuna delle classi restituite dal TransferManagerupload*,download*, ecopymethods restituisce un'istanza di una delle seguenti classi, a seconda che si tratti di un'operazione a file singolo o a più file.

Classe	Restituiti da
Copia	copy
Scarica	download
MultipleFileDownload	downloadDirectory

Classe	Restituiti da
Caricamento	`upload`
MultipleFileUpload	`uploadFileList` , `uploadDirectory`

Tutte queste classi implementano iltrasferimentointerfaccia.`Transfer`fornisce metodi utili per ottenere l'avanzamento di un trasferimento, sospendere o riprendere il trasferimento e ottenere lo stato corrente o finale del trasferimento.

Argomenti

- Sondaggio dell'attuale avanzamento di un trasferimento
- Ottieni l'avanzamento del trasferimento con un ProgressListener
- Ottieni lo stato di avanzamento dei subtrasferimenti

Sondaggio dell'attuale avanzamento di un trasferimento

Questo ciclo stampa lo stato di avanzamento di un trasferimento, ne esamina il progresso 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 l'avanzamento del trasferimento con un ProgressListener

È possibile collegare unProgressListenera qualsiasi trasferimento utilizzando iltrasferimentointerfacciaaddProgressListenermetodo.

UNProgressListenerrichiede un solo metodo,progressChanged, che richiede unProgressEventoggetto. È possibile utilizzare l'oggetto per ottenere i byte totali dell'operazione chiamandonegetBytesmetodo e il numero di byte trasferiti finora chiamandogetBytesTransferred.

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

LaMultipleFileUploadclass può restituire informazioni sui suoi subtrasferimenti chiamandoli`getSubTransfers`metodo. Restituisce un elemento non modificabileRaccoltadiCaricamentooggetti che forniscono lo stato di trasferimento individuale e l'avanzamento di ogni subtrasferimento.

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

- Chiavi degli oggettinellaAmazon Simple Storage ServiceGuida per l'utente di

Configurazione di unAmazon S3Bucket come sito web

È possibile configurare unAmazon S3per comportarsi come un sito Web. Per fare ciò, è necessario impostare la configurazione del sito Web.

> **Note**
>
> Questi esempi di codice presuppongono che tu comprenda il materiale inUtilizzo diAWS SDK for Javae hanno configurato il valore predefinitoAWScredenziali che utilizzano le informazioni inConfigurazioneAWSCredenziali e regione per lo sviluppo.

Impostare la configurazione del sito Web di un bucket

Per impostare unAmazon S3configurazione del sito web di bucket, chiama AmazonS3`setWebsiteConfiguration`metodo con il nome del bucket per impostare la configurazione e a`BucketWebsiteConfiguration`oggetto contenente la configurazione del sito Web del bucket.

L'impostazione di un documento indice ènecessario; tutti gli altri parametri sono opzionali.

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 di una configurazione del sito Web non modifica le autorizzazioni di accesso per il bucket. Per rendere visibili i tuoi file sul Web, dovrai anche impostare unpolicy di bucketche consente l'accesso pubblico in lettura ai file nel bucket. Per ulteriori informazioni, consultaGestione degli accessi aAmazon S3Periodi di bucket che utilizzano policy di bu.

Vedi l'esempio completo su GitHub.

Ottieni la configurazione del sito Web di un Bucket

Per ottenere unAmazon S3configurazione del sito web di bucket, chiama AmazonS3`getWebsiteConfiguration`con il nome del bucket per il quale recuperare la configurazione.

La configurazione verrà restituita come aBucketWebsiteConfigurationoggetto. Se non esiste una configurazione del sito Web per il bucket, allora`null`verrà restituito.

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.

Eliminare la configurazione del sito Web di un bucket

Per eliminare unAmazon S3configurazione del sito web di bucket, chiama AmazonS3`deleteWebsiteConfiguration`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);
}
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- PUT Bucket websitenellaAmazon S3Documentazione di riferimento API
- GET Bucket websitenellaAmazon S3Documentazione di riferimento API
- DELETE Bucket websitenellaAmazon S3Documentazione di riferimento API

UtilizzaAmazon S3Crittografia lato client di

Crittografia dei dati tramiteAmazon S3il client di crittografia è uno dei modi in cui è possibile fornire un ulteriore livello di protezione per le informazioni sensibili in cui memorizziAmazon S3. Gli esempi in questa sezione mostrano come creare e configurare laAmazon S3client di crittografia per la tua applicazione.

Se non si è utilizzato per la crittografia, consulta la sezioneNozioni di base sulla crittografianellaAWSKMS Developer Guide per una panoramica di base dei termini e degli algoritmi di crittografia. Per informazioni sul supporto crittografico in tuttoAWSSDK, consultaAWSSupporto degli SDK perAmazon S3Crittografia lato clientnellaAmazon Web ServicesRiferimenti generali.

> **Note**
>
> Questi esempi di codice presuppongono che tu comprenda il materiale inUtilizzo diAWS SDK for Javae hanno configurato il valore predefinitoAWScredenziali che utilizzano le informazioni inConfigurazioneAWSCredenziali e regione per lo sviluppo.

Se si utilizza la versione 1.11.836 o precedente delAWS SDK for Java, consultaAmazon S3Crittografia della migrazioneper informazioni sulla migrazione delle applicazioni a versioni successive. Se non è possibile eseguire la migrazione, vederequesto esempio completosu GitHub

In caso contrario, se si utilizza la versione 1.11.837 o le versioni successive delAWS SDK for Java, esplora gli argomenti di esempio elencati di seguito da utilizzareAmazon S3crittografia lato client.

Argomenti

- Amazon S3crittografia lato client con chiavi master client
- Amazon S3crittografia lato client conAWSChiavi gestite KMS

Amazon S3crittografia lato client con chiavi master client

I seguenti esempi utilizzano laBuilder V2 Client di crittografia Amazon S3per creare una classeAmazon S3client con crittografia lato client abilitata. Una volta abilitato, tutti gli oggetti su cui carichiAmazon S3l'utilizzo di questo client sarà crittografato. Qualsiasi oggetto da cui si ottieneAmazon S3l'utilizzo di questo client verrà automaticamente decrittografato.

> **Note**
>
> I seguenti esempi mostrano l'utilizzo diAmazon S3Crittografia lato client con chiavi master client gestite dal cliente. Per informazioni sull'utilizzo della crittografia conAWSChiavi gestite KMS, vediAmazon S3crittografia lato client conAWSChiavi gestite KMS.

È possibile scegliere tra due modalità di crittografia quando si abilita lato clientAmazon S3crittografia: autenticata o autenticata rigorosa. Nelle sezioni seguenti viene illustrato come abilitare ciascun tipo. Per informazioni sugli algoritmi utilizzati da ciascuna modalità, consulta ilModalità CryptoModedefinizione.

Importazioni richieste

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 rigor

La crittografia autenticata rigorosa è la modalità predefinita se no`CryptoMode`è specificato.

Per abilitare esplicitamente questa modalità, specificare il`StrictAuthenticatedEncryption`nel`withCryptoConfiguration`metodo.

> **Note**
>
> Per utilizzare la crittografia autenticata lato client, devi includere la versione più recenteBouncy Castle jarfile nel classpath della tua 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 crittografia autenticata

Quando utilizzi`AuthenticatedEncryption`modalità, un algoritmo di key wrapping migliorato viene applicato durante la crittografia. Durante la decrittografia in questa modalità, l'algoritmo può verificare l'integrità dell'oggetto decrittografato e generare un'eccezione se il controllo non riesce. Per ulteriori informazioni su come funziona la crittografia autenticata, consulta la sezioneAmazon S3Crittografia autenticata lato clientpost di blog.

> **Note**
>
> Per utilizzare la crittografia autenticata lato client, devi includere la versione più recenteBouncy Castle jarfile nel classpath della tua applicazione.

Per abilitare questa modalità, specificare il`AuthenticatedEncryption`nel`withCryptoConfiguration`metodo.

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 S3crittografia lato client conAWSChiavi gestite KMS

Negli esempi seguenti viene utilizzato ilBuilder V2 Client di crittografia Amazon S3classe per creare unaAmazon S3client con crittografia lato client abilitata. Una volta configurato, tutti gli oggetti su cui carichiAmazon S3l'utilizzo di questo client sarà crittografato. Qualsiasi oggetto da cui ottieniAmazon S3l'utilizzo di questo client viene decrittografato automaticamente.

> **Note**
>
> Negli esempi seguenti viene illustrato come utilizzare laAmazon S3crittografia lato client conAWSChiavi gestite KMS. Per informazioni su come usare la crittografia con le tue chiavi, consultaAmazon S3crittografia lato client con chiavi master client.

È possibile scegliere tra due modalità di crittografia quando si abilita lato clientAmazon S3crittografia: autenticata o autenticata rigorosa. Nelle sezioni seguenti viene illustrato come abilitare ciascun tipo. Per sapere quali algoritmi utilizzano ciascuna modalità, vedere ilCryptoMode crittografiadefinizione.

Importazioni richieste

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

La crittografia autenticata rigorosa è la modalità predefinita in caso contrario`CryptoMode`è specificato.

Per abilitare esplicitamente questa modalità, specificare il`StrictAuthenticatedEncryption`valore nel`withCryptoConfiguration`metodo.

> **Note**
>
> Per usare la crittografia autenticata lato client, è necessario includere la versione più recenteBouncy Castle jarfile nel classpath della tua 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));
```

Chiama il`putObject`sul metodoAmazon S3client di crittografia per caricare oggetti.

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 memorizzata.

Codice

```
System.out.println(s3Encryption.getObjectAsString(bucket_name, ENCRYPTED_KEY3));
```

Modalità crittografia autenticata

Quando utilizzi`AuthenticatedEncryption`modalità, un algoritmo di key wrapping migliorato viene applicato durante la crittografia. Durante la decrittografia in questa modalità, l'algoritmo può verificare l'integrità dell'oggetto decrittografato e generare un'eccezione se il controllo non riesce. Per ulteriori informazioni su come funziona la crittografia autenticata, consulta la sezioneAmazon S3Crittografia autenticata lato clientpost di blog.

> **Note**
>
> Per usare la crittografia autenticata lato client, è necessario includere la versione più recenteBouncy Castle jarfile nel classpath della tua applicazione.

Per abilitare questa modalità, specificare il`AuthenticatedEncryption`valore nel`withCryptoConfiguration`metodo.

Codice

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

Configurazione dellaAWS KMScliente

LaAmazon S3client di crittografia crea unAWS KMSclient per impostazione predefinita, a meno che non sia specificato esplicitamente.

Per impostare la regione per questo creato automaticamenteAWS KMSclient, imposta il`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 la propriaAWS KMSclient 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();
```

Esempi di Amazon SQS utilizzando l'AWS SDK for Java

In questa sezione vengono forniti esempi di programmazione di Amazon SQS utilizzando AWS SDK for Java.

> **Note**
>
> Gli esempi includono solo il codice necessario per dimostrare ciascuna 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 diAmazon SQSCode di messaggi

Una coda di messaggi è il container logico utilizzato per l'invio di messaggi in modo affidabile in 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 laAmazon SQSGuida per gli sviluppatori.

In questo argomento viene descritto come creare, elencare, eliminare e ottenere l'URL di una coda Amazon SQS utilizzando AWS SDK for Java.

Creare una coda

Usa il client Amazon`SQS`createQueuemetodo, fornendo unCreateQueueRequestl'oggetto 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;
    }
}
```

È possibile 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.

Elencare code

Per elencareAmazon SQScode per l'account, chiama il client Amazon SQS`listQueues`metodo.

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);
}
```

Utilizzo di`listQueues`sovraccarico senza alcun parametro restituiscetutte le code. È possibile filtrare i risultati restituiti passandoli a`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 client AmazonSQS`getQueueUrl`metodo.

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();
```

Vedi l'esempio completo su GitHub.

Eliminare una coda

Fornisci la codaURLal client AmazonSQS`deleteQueue`metodo.

Importazioni

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

Codice

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

```
sqs.deleteQueue(queue_url);
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- ComeAmazon SQSFunzionano le codenellaAmazon SQSGuida per gli sviluppatori
- CreateQueuenellaAmazon SQSDocumentazione di riferimento API
- GetQueueUrlnellaAmazon SQSDocumentazione di riferimento API
- ListQueuesnellaAmazon SQSDocumentazione di riferimento API
- DeleteQueuesnellaAmazon SQSDocumentazione di riferimento API

Invio, ricezione ed eliminazioneAmazon SQSMessaggi

In questo argomento viene descritto l'invio, ricezione ed eliminazioneAmazon SQSMessaggi. I messaggi vengono sempre distribuiti tramite una coda SQS.

Inviare un messaggio

Aggiungi un solo messaggio a un oggettoAmazon SQScoda chiamando il client AmazonSQS`sendMessage`Metodo. Fornisci un oggetto SendMessageRequest contenente 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 messaggi in una singola richiesta. Per inviare più messaggi, utilizza il client Amazon SQS `sendMessageBatch` metodo, che richiede un SendMessageBatchRequest contenente l'URL della coda e un elenco di messaggi (ognuno di SendMessageBatchRequestEntry) da inviare. È inoltre possibile 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 eventuali messaggi che si trovano attualmente nella coda chiamando il client Amazon SQS `receiveMessage` Metodo, passando 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 messaggi dopo la ricezione

Dopo aver ricevuto un messaggio ed elaborato il relativo contenuto, elimina il messaggio dalla coda inviando l'handle di ricezione del messaggio e l'URL della coda al client AmazonSQS`deleteMessage`Metodo.

Codice

```
for (Message m : messages) {
    sqs.deleteMessage(queueUrl, m.getReceiptHandle());
}
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- ComeAmazon SQSUtilizzo delle codenellaAmazon SQSGuida per gli sviluppatori
- SendMessagenellaAmazon SQSDocumentazione di riferimento API
- SendMessageBatchnellaAmazon SQSDocumentazione di riferimento API
- ReceiveMessagenellaAmazon SQSDocumentazione di riferimento API
- DeleteMessagenellaAmazon SQSDocumentazione di riferimento API

Abilitazione del long polling perAmazon SQSCoda di messaggi

Amazon SQSutilizzashort pollingper impostazione predefinita, eseguendo query solo a un sottoinsieme di server, in base a una distribuzione random ponderata, per determinare se i messaggi sono disponibili per essere inclusi nella risposta.

Il polling lungo aiuta a ridurre i costi di utilizzoAmazon SQSriducendo il numero di risposte vuote quando non ci sono messaggi disponibili da restituire in risposta a ReceiveMessage richiesta inviata a unAmazon SQScoda ed eliminazione di false risposte vuote.

> **Note**
>
> È possibile impostare una frequenza di polling lunga da1-20 secondi.

Abilitazione del long polling alla creazione di una coda

Per abilitare il polling lungo durante la creazione di unAmazon SQScoda, imposta il`ReceiveMessageWaitTimeSeconds`attributo sulCreateQueueRequestoggetto prima di chiamare la classe AmazonSQS»`createQueue`metodo.

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;
    }
}
```

Vedi 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, è anche possibile attivarla su una coda esistente impostando`ReceiveMessageWaitTimeSeconds`sulSetQueueAttributesRequestprima di chiamare la classe AmazonSQS»`setQueueAttributes`metodo.

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);
```

Vedi l'esempio completo su GitHub.

Abilitazione del long polling alla ricezione del messaggio

È possibile abilitare il polling lungo quando si riceve un messaggio impostando il tempo di attesa in secondi sulReceiveMessageRequestche fornisci alla classe AmazonSQS»`receiveMessage`metodo.

> **Note**
>
> Dovresti assicurarti che ilAWSIl timeout della richiesta del cliente è superiore al tempo massimo lungo di sondaggio (20s) in modo che il tuo`receiveMessage`le richieste non scadono durante l'attesa del prossimo evento del sondaggio!

Importazioni

```
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
```

Codice

```
ReceiveMessageRequest receive_request = new ReceiveMessageRequest()
        .withQueueUrl(queue_url)
        .withWaitTimeSeconds(20);
sqs.receiveMessage(receive_request);
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Amazon SQSLong PollingnellaAmazon SQSGuida per gli sviluppatori

- CreateQueuenellaAmazon SQSDocumentazione di riferimento API
- ReceiveMessagenellaAmazon SQSDocumentazione di riferimento API
- SetQueueAttributesnellaAmazon SQSDocumentazione di riferimento API

Impostare il timeout visibilità inAmazon SQS

Quando viene ricevuto un messaggio inAmazon 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 datoTimeout visibilitàper evitare che il messaggio venga ricevuto più di una volta prima che possa essere elaborato ed eliminato.

> **Note**
>
> Quando utilizzicode standardil timeout visibilità non garantisce che non venga ricevuto un messaggio due volte. Se stai utilizzando una coda standard, assicurati che il codice sia in grado di gestire il caso in cui lo stesso messaggio è stato recapitato più di una volta.

Impostazione del timeout visibilità per un messaggio singolo

Dopo aver ricevuto un messaggio, è possibile modificarne il timeout di visibilità passando il quadratino di ricezione in unChangeMessageVisibilityRequestche passi alla classe AmazonSQS»`changeMessageVisibility`Metodo.

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);
```

Vedi 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, creare un elenco diChangeMessageVisibilityBatchRequestEntryoggetti, ognuno contenente una stringa ID univoca e un handle di ricevuta. Quindi, passa l'elenco alAmazon SQSclasse client'`changeMessageVisibilityBatch`Metodo.

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);
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Timeout visibilitànellaAmazon SQSGuida per gli sviluppatori
- SetQueueAttributesnellaAmazon SQSDocumentazione di riferimento API
- GetQueueAttributesnellaAmazon SQSDocumentazione di riferimento API
- ReceiveMessagenellaAmazon SQSDocumentazione di riferimento API
- ChangeMessageVisibilitynellaAmazon SQSDocumentazione di riferimento API
- ChangeMessageVisibilityBatchnellaAmazon SQSDocumentazione di riferimento API

Utilizzo di code DLQ in Amazon SQS

Amazon SQSfornisce supporto percode a lettere morte. Una coda a cui altre code (origini) possono mirare 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 a lettere morte

Una coda con lettere morte viene creata allo stesso modo di una coda normale, ma ha le seguenti restrizioni:

- Una coda con lettere morte deve essere dello stesso tipo di coda (FIFO o standard) della coda di origine.
- È necessario creare una coda con lettere morte usando la stessaAccount AWSe regione come coda di origine.

Qui creiamo due identiciAmazon SQScode, una delle quali servirà come 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 a lettere morte per una coda sorgente

Per designare una coda con lettere morte, è necessario innanzitutto creare unpolitica di redrivee quindi imposta il criterio negli attributi della coda. Un criterio di redrive è specificato 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 che venga inviato alla coda delle lettere morte.

Per impostare il criterio di redrive per la coda di origine, chiama la classe AmazonSQS»`setQueueAttributes`metodo con unSetQueueAttributesRequestoggetto per il quale hai impostato il`RedrivePolicy`attributo con la tua politica di redrive JSON.

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);
```

Vedi l'esempio completo su GitHub.

Ulteriori informazioni

- Utilizzo diAmazon SQSCode DLQnellaAmazon SQSGuida per gli sviluppatori
- SetQueueAttributesnellaAmazon SQSDocumentazione di riferimento API

Esempi di Amazon SWF utilizzando l'AWS SDK for Java

Amazon SWFè un servizio di gestione dei flussi di lavoro che aiuta gli sviluppatori a creare e scalare flussi di lavoro distribuiti che possono avere passaggi paralleli o sequenziali consistenti in attività, flussi di lavoro figlio o addirittura Lambdacompiti.

Ci sono due modi per lavorare conAmazon SWFutilizzandoAWS SDK for Java, utilizzando il SWFclienteoggetto o usando ilAWS Flow Frameworkper Java. LaAWS Flow Frameworkper Java è

più difficile da configurare inizialmente, poiché fa uso pesante delle annotazioni e si basa su librerie aggiuntive come AspecTJ e Spring Framework. Tuttavia, per progetti grandi o complessi, risparmierai tempo di codifica utilizzando ilAWS Flow Frameworkper Java. Per ulteriori informazioni, consulta la .AWS Flow Frameworkper sviluppatori Java.

In questa sezione vengono forniti esempi di programmazioneAmazon SWFusando ilAWS SDK for Javadirettamente client.

Argomenti

Nozioni di base su SWF

Questi sono modelli generali per lavorare conAmazon SWFutilizzando ilAWS SDK for Java. È inteso principalmente per riferimento. Per un tutorial introduttivo più completo, consulta la paginaCostruire un SempliciAmazon SWFApplicazione.

Dipendenze

BaseAmazon SWFle applicazioni richiederanno le seguenti dipendenze, incluse nelAWS SDK for Java:

- aws-java-sdk-1.12.*.jar
- commons-logging-1.2.*.jar
- http://client-4.3.*.jar
- http://core-4.3.*.jar
- jackson-annotations-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 disponibile, ma le versioni fornite con l'SDK sono state testate per la compatibilità e sono quelle da utilizzare.

AWS Flow Frameworkper le applicazioni Java richiede una configurazione aggiuntiva,edipendenze aggiuntive. Consultare ilAWS Flow FrameworkGuida per sviluppatori di Javaper ulteriori informazioni sull'utilizzo del framework.

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 di cui hai bisogno. Probabilmente finirai per specificare classi particolari nelcom.amazonaws.services.simpleworkflow.modelworkspace:

```
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 ilAWS Flow Frameworkper Java, importerai le classi dalcom.amazonaws.services.simpleworkflow.flowworkspace. Ad esempio:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.flow.ActivityWorker;
```

> **Note**
>
> LaAWS Flow Frameworkper Java ha requisiti aggiuntivi oltre a quelli della baseAWS SDK for Java. Per ulteriori informazioni, consulta la .AWS Flow FrameworkGuida per sviluppatori di Java.

Utilizzo della classe client SWF

La tua interfaccia di base perAmazon SWFè attraverso o ilAmazonSimpleWorkflowClientoAmazonSimpleWorkflowAsyncClientlezioni. La principale differenza tra questi è che il*AsyncClientritorno classeFutureoggetti per la programmazione simultanea (asincrona).

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

Creazione di un'Amazon SWFapplicazione semplice

Questo argomento ti introdurrà alla programmazione di Amazon SWFapplicazioni conAWS SDK for Java, presentando nel contempo alcuni concetti importanti.

Informazioni sull'esempio

Il progetto di esempio creerà un flusso di lavoro con una singola attività che accetta i dati del flusso di lavoro passati attraverso il AWS cloud (nella tradizione diHelloWorld, sarà il nome di qualcuno da salutare) e quindi stampa un saluto in risposta.

Sebbene questo sembri molto semplice in apparenza, 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 del codice che definiscono l'ordine logico di esecuzione delle attività del flusso di lavoro e dei flussi di lavoro secondari.
- Un operatore del flusso di lavoro, noto anche come decisore, che effettua sondaggi per le attività decisionali e pianifica le attività o i flussi di lavoro dei figli in risposta.
- Una o più attività, ognuna delle quali rappresenta un'unità di lavoro nel flusso di lavoro.
- Un operatore delle attività che effettua sondaggi per le attività e esegue i metodi di attività in risposta.
- Uno o più elenchi di attività, ovvero code gestite e Amazon SWF utilizzate per inviare richieste ai lavoratori del flusso di lavoro e delle attività. Le attività in un elenco di attività destinate ai lavoratori del flusso di lavoro sono chiamate attività decisionali. Quelli destinati agli addetti alle attività sono chiamati compiti di attività.
- Uno strumento per avviare il 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, passa i dati tra di loro, gestisce i timeout e le notifiche del battito cardiaco 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 for Java.
- Apache Maven.
- JDK 1.7 o versioni successive. 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 proseguire. Ulteriori informazioni sulla configurazione e l'utilizzo di AWS SDK for Java con vari sistemi di build sono disponibili in Guida introduttiva.
> Allo stesso modo, ma con maggiore impegno, i passaggi mostrati qui possono essere implementati utilizzando uno qualsiasi degli AWS SDK con supporto perAmazon SWF.

Tutte le dipendenze esterne necessarie sono incluse inAWS SDK for Java, quindi non c'è nulla di aggiuntivo da scaricare.

AWSAccesso

Per eseguire correttamente questo tutorial, è necessario accedere al portale di AWS accesso come descritto nella sezione relativa alla 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 con le autorizzazioni per accedere ad Amazon SWF. AWS IAM Identity Center Dopo aver incollato le credenziali temporanee, il `credentials` file risulterà 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 `default` profilo.

Creazione di un progetto SWF

1. Iniziare 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` cartella e tutto ciò che contiene, non la useremo per questo tutorial. Puoi anche eliminare`App.java`, poiché lo sostituiremo con nuove classi.
2. Modifica il `pom.xml` file del progetto e aggiungi il aws-java-sdk-simpleworkflowmodulo aggiungendo 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 `<dependencies>` blocco) in`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 visiteremo una per una:

- HelloTypes.java --contiene i dati del dominio, dell'attività e del 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 effettua sondaggi per le attività ed esegue le attività in risposta.
- WorkflowWorker.java --contiene il workflow worker (decider), che effettua sondaggi per le 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 flusso di lavoro che i dipendenti possono utilizzare.

Passaggi comuni per tutti i file sorgente

Tutti i file che crei per ospitare le tue classi Java avranno alcune cose in comune. Per risparmiare tempo, ogni volta che aggiungerete un nuovo file al progetto, saranno necessari questi passaggi:

1. Crea il file nella `src/main/java/aws/example/helloswf/` directory 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.*;
```

Registrazione di un dominio, un flusso di lavoro e un tipo di attività

Inizieremo creando una nuova classe eseguibile,`HelloTypes.java`. Questo file conterrà dati condivisi che le diverse parti del flusso di lavoro devono conoscere, come il nome e la versione dell'attività e dei tipi di flusso di lavoro, il nome di dominio e il nome dell'elenco delle attività.

1. Apri il tuo editor di testo e crea il file`HelloTypes.java`, aggiungi una dichiarazione di pacchetto e importa secondo i passaggi più 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 daAWS SDK for 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 usi un profilo diverso, modifica il codice precedente come segue e sostituisci *`profile_name con il nome`* del nome effettivo del profilo.

```
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 flussi di lavoro 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 registri un dominio, gli fornisci un nome (qualsiasi set da 1 a 256 caratteri esclusi`:`,, `/`|, i caratteri di controllo o la stringa letterale '`arn') e un periodo di conservazione, che è 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 massimo è di 90 giorni. RegisterDomainRequestPer ulteriori informazioni, consulta.

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 for Java metodi, i dati per il metodo sono forniti da una classe nello `simpleworkflow.model`

spazio dei nomi, 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 nel dominio in cui è registrata. Le attività contengono anche una serie di parametri opzionali, come l'elenco delle attività predefinito utilizzato per ricevere attività e dati da SWF e una serie di timeout diversi che è possibile utilizzare per porre vincoli sulla durata delle diverse parti dell'esecuzione dell'attività. RegisterActivityTypeRequestPer ulteriori informazioni, consulta.

> **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, TypeAlreadyExistsExceptionviene generato un. 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 una versione e dispongono anche di timeout configurabili. RegisterWorkflowTypeRequestPer ulteriori informazioni, consulta.

+

Se il tipo di flusso di lavoro che stai cercando di registrare esiste già, TypeAlreadyExistsExceptionviene generato un. Infine, rendi la classe eseguibile fornendole un `main` metodo che registrerà a sua volta 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 con la codifica degli addetti all'attività e al flusso di lavoro. Una volta registrati il dominio, il flusso di lavoro e l'attività, non sarà necessario eseguirli nuovamente: questi tipi persistono finché non li dichiarerai obsoleti tu stesso.

Implementa l'operatore delle attività

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 flusso di lavoro tipico è in genere costituito da una serie di attività che possono essere eseguite in modo sincrono, asincrono o una combinazione di entrambi.

L'activity worker è la parte di codice che analizza le attività generate Amazon SWF in risposta alle decisioni del 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 operatore che gestisca una singola attività.

1. Apri il tuo editor di testo e crea il file`ActivityWorker.java`, aggiungi una dichiarazione di pacchetto e importa secondo i passaggi più 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. Aggiungi la `ActivityWorker` classe al file e assegnagli un membro dei dati per contenere un client SWF con Amazon SWF cui interagire:

```
    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 sondare 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 passaggio. PollForActivityTaskRequest

Una volta ricevuta un'attività, ne riceviamo un identificatore univoco chiamando il metodo dell'`getTaskToken`attività.

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.

```
    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`, possiamo iniziare a eseguire il metodo activity (`sayHello`), fornendogli i dati di input inviati con l'attività.

Se l'operazione è riuscita (non è stato generato alcun errore), il worker 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'attività non è riuscita, rispondiamo chiamando il `respondActivityTaskFailed` metodo con un RespondActivityTaskFailedRequestoggetto, passandogli il token dell'attività e le informazioni sull'errore.

> **Note**
>
> Questa attività non si interromperà correttamente se viene uccisa. Sebbene non rientri nello scopo di questo tutorial, un'implementazione alternativa di questo activity worker è fornita nell'argomento di accompagnamento, Shutting Down Activity and Workflow Workers con garbo.

Impilare l'operatore del flusso di lavoro

La logica del flusso di lavoro risiede in un pezzo di codice noto come workflow worker. Il workflow worker effettua un sondaggio per le attività decisionali inviate dal dominio e dall'elenco di attività predefinito Amazon SWF in cui è stato registrato il tipo di flusso di lavoro.

Quando il lavoratore del 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 il tuo editor di testo e crea il file`WorkflowWorker.java`, aggiungi una dichiarazione di pacchetto e importa secondo i passaggi più 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. Aggiungi il `main` metodo. Il metodo si ripete continuamente, esaminando 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, chiamiamo il suo `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 task token e l'elenco degli HistoryEventoggetti inviati con l'attività.

5. Aggiungi 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 impostato 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 molto 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, è la parte relativa al nome del nostro saluto, quindi viene salvata in una stringa da utilizzare quando si pianifica 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 solo un'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 usando una versione precedente di Java, puoi utilizzare la EventTypeclasse per convertire il valore `String` restituito `history_event.getType()` da in un valore enum e poi di nuovo in un, `String` se necessario:

```
EventType et = EventType.fromValue(event.getEventType());
```

1. Dopo la `switch` dichiarazione, aggiungi 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 ancora stata programmata, rispondiamo con una `ScheduleActivityTask` decisione, che fornisce informazioni in una ScheduleActivityTaskDecisionAttributesstruttura sull'attività da programmare successiva, compresi 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 dell'operazione che identifica l'operazione e l'elenco di `Decision` oggetti.

Implementazione del flusso di lavoro

Infine, scriveremo del codice per avviare l'esecuzione del flusso di lavoro.

1. Apri il tuo editor di testo e crea il file`WorkflowStarter.java`, aggiungi una dichiarazione di pacchetto e importa secondo i passaggi più 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 singolo metodo`main`, che accetta un argomento opzionale passato alla 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 flusso di lavoro 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, che l'intero flusso di lavoro dovrebbe impiegare per l'esecuzione.

L'oggetto Run che `startWorkflowExecution` restituisce fornisce un ID di esecuzione, un valore che può essere utilizzato per identificare l'esecuzione di questo particolare flusso di lavoro nella Amazon SWF cronologia delle esecuzioni del flusso di lavoro.

+ NOTA: l'ID di esecuzione è generato da Amazon SWF e non è uguale al nome di esecuzione del flusso di lavoro che fornisci all'avvio dell'esecuzione del flusso di lavoro.

Compilare 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 `target` directory.

Esecuzione dell'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 si utilizza Windows, è necessario aprire due istanze della riga di comando aggiuntive e accedere alla `helloswf` directory contenuta in ciascuna di esse.

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 sue dipendenze dal tuo classpath Java. Puoi impostare la variabile di `CLASSPATH` ambiente sulla posizione delle tue librerie AWS SDK e sulla `third-party/lib` directory nell'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 ogni 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 creare il codice, provate entrambi a eseguire gli esempi e ottenete una serie di errori "NoClassDefFound", probabilmente perché il classpath è impostato in modo errato.

Registrazione del tipo di attività,

Prima di gestire i tuoi worker e il workflow starter, dovrai registrare il dominio e il tuo flusso di lavoro e i tipi di attività. Il codice per eseguire questa operazione è stato implementato in Registra un dominio, flusso 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 eseguire il sondaggio delle attività fino a quando non verranno eliminate, quindi dovresti eseguirle in finestre di terminale separate oppure, se stai usando 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.

Avvia l'esecuzione del flusso di lavoro

Ora che gli addetti all'attività e al flusso di lavoro stanno effettuando un sondaggio, puoi avviare l'esecuzione del flusso di lavoro. Questo processo verrà eseguito fino a quando il flusso di lavoro non

restituirà lo stato completato. Dovresti eseguirlo in una nuova finestra di terminale (a meno che tu non esegua i tuoi lavoratori 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. Ad esempio:
>
> ```
> echo "## Running $className..."
> ```

Una volta avviata l'esecuzione del flusso di lavoro, dovresti iniziare a vedere l'output fornito da entrambi i lavoratori e dall'esecuzione del flusso di lavoro stesso. Al termine del flusso di lavoro, 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 correttamente i lavoratori, vedi Chiusura regolare dei lavoratori di Activity and Workflow Workflow.
- Per ulteriori informazioniAmazon SWF, visita la Amazon SWFhome page o visualizza la Guida per gli Amazon SWF sviluppatori.
- 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 Guida AWS Flow Framework per gli sviluppatori di Java.

Task di Lambda

In alternativa o in combinazione con,Amazon SWFattività, è possibile utilizzareLambdafunzioni per rappresentare le unità di lavoro nei flussi di lavoro e pianificarle in modo simile alle attività.

Questo argomento si concentra su come implementareAmazon SWF Lambdaattività che utilizzano ilAWS SDK for Java. Per ulteriori informazioni suLambdaattività in generale, vediAWS LambdaAttivitànellaAmazon SWFGuida per sviluppatori di .

Configurare un ruolo IAM cross-service per eseguire la funzione Lambda

PrimaAmazon SWFpuò eseguire il tuoLambda, è necessario impostare un ruolo IAM da assegnareAmazon SWFpermesso di eseguireLambdafunziona automaticamente. Per informazioni complete su come eseguire questa operazione, consultaAWS LambdaAttività.

Quando registri un flusso di lavoro utilizzato, avrai bisogno dell'ARN (Amazon Resource Name) di questo ruolo IAM.Lambdaattività.

Creazione di una funzione Lambda

È possibile scrivereLambdafunzioni in diverse lingue, incluso Java. Per informazioni complete su come creare, distribuire e utilizzareLambdafunzioni, consultaAWS LambdaGuida per gli sviluppatori.

> **Note**
>
> Non importa quale lingua usi per scrivereLambdafunzione, può essere pianificata e gestita daqualsiasi Amazon SWFindipendentemente dalla lingua in cui è scritto il codice del flusso di lavoro.Amazon SWFgestisce i dettagli dell'esecuzione della funzione e del passaggio di dati da e verso di essa.

Ecco un sempliceLambdafunzione che potrebbe essere utilizzata al posto dell'attività inCostruire un SempliciAmazon SWFApplicazione.

- Questa versione è scritta in JavaScript, che può essere inserita direttamente utilizzando ilAWS Management Console:

```
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, consultaCreazione di un pacchetto di distribuzione (Java)nellaAWS LambdaGuida per sviluppatori di . Dovrai anche guardare la sezione intitolataModello di programmazione per la creazioneLambdaFunzioni in Java.

Lambdale funzioni prendono uneventooingressooggetto come primo parametro e acontestooggetto come secondo, che fornisce informazioni sulla richiesta di esecuzioneLambdafunzione. Questa particolare funzione prevede che l'input sia in JSON, con unwhocampo impostato sul nome utilizzato per creare il saluto.

Registra un flusso di lavoro da utilizzare con Lambda

Per un flusso di lavoro per pianificare unLambda, è necessario specificare il nome del ruolo IAM fornitoAmazon SWFcon l'autorizzazione a invocareLambdaFunzioni . È possibile impostarlo durante la registrazione del flusso di lavoro utilizzando il`withDefaultLambdaRole`o`setDefaultLambdaRole`metodi diRegisterWorkflowTypeRequest.

```
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) {
```

Pianificare unLambdacompito

Pianificare unLambdaè simile alla pianificazione di un'attività. Fornire unDecisionecon una funzione `Schedulelambda`DecisionTypee conScheduleLambdaFunctionDecisionAttributes LAMBDA.

```
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(
```

Nella`ScheduleLambdaFuntionDecisionAttributes`, è necessario fornire unnome, che è l'ARN dellLambdafunzione da chiamare e unid, che è il nomeAmazon SWFutilizzerà per identificare illLambdafunzione nei registri della cronologia.

È anche possibile definire opzionaliingressoperLambdafunzione e imposta il suoinizia a chiudere il timeoutvalore, che è il numero di secondi che ilLambdaè consentita l'esecuzione prima di generare un`LambdaFunctionTimedOut`Evento .

> **Note**
>
> Questo codice utilizza ilCliente AWS Lambdaper recuperare l'ARN delLambda, dato il nome della funzione. È possibile utilizzare questa tecnica per evitare di codificare l'ARN completo (che include ilAccount AWSID) nel tuo codice.

Gestisci gli eventi della funzione Lambda nel tuo decider

Lambdale attività genereranno una serie di eventi su cui è possibile intervenire durante il polling per le attività decisionali nel workworker del flusso di lavoro, corrispondenti al ciclo di vita delLambdacompito, conTipoeventovalori come`LambdaFunctionScheduled`,`LambdaFunctionStarted`, e`LambdaFunctionCompleted`. Se il fileLambdala funzione non riesce o richiede più tempo per essere eseguita rispetto al valore di timeout impostato, si riceverà un`LambdaFunctionFailed`o`LambdaFunctionTimedOut`tipo di evento, rispettivamente.

```
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 output dal tuoLambdafunzione

Quando ricevi unLambdaFunctionCompleted`EventType, you can retrieve your 0 function's return value by first calling `getLambdaFunctionCompletedEventAttributessulHistoryEventper ottenere unLambdaFunctionCompletedEventAttributes Lambdaoggetto, e quindi chiamandogetResultmetodo per recuperare l'output delLambdafunzione:

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

Fonte completa per questo esempio

È possibile sfogliarefonte completa: github: `<awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf_lambda/>per questo esempio su Github nelguida per sviluppatori aws-java-repository.

Chiudere con grazia i lavoratori delle attività e del flusso di lavoro

LaCostruire un SempliciAmazon SWFApplicazionetopic ha fornito un'implementazione completa di una semplice applicazione di flusso di lavoro composta da un'applicazione di registrazione, un lavoratore di attività e flusso di lavoro e uno starter del flusso di lavoro.

Le classi di lavoro sono progettate per essere eseguite continuamente, polling per le attività inviate daAmazon SWFper gestire attività o decisioni di rimpatrio. Una volta effettuata una richiesta di sondaggio,Amazon SWFregistra il poller e tenterà di assegnargli un'attività.

Se il lavoratore del flusso di lavoro viene terminato durante un lungo sondaggio,Amazon SWFpotrebbe comunque provare a inviare un'attività al lavoratore terminato, con conseguente perdita di un'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 termina.

In questo argomento, riscriveremo il lavoratore dell'attività da`helloswf`, utilizzando gli hook di arresto di Java per tentare un arresto aggraziato del lavoratore dell'attività.

Di seguito è riprodotto 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 nel`main`la funzione nella versione originale è stata spostata nel proprio metodo,`pollAndExecute`.

La`main`la funzione ora utilizza aCountDownLatchin combinazione con ungancio di arrestoper far sì che il thread attenda fino a 60 secondi dopo la sua richiesta di terminazione prima di chiudere il thread.

Registrazione di domini

Ogni flusso di lavoro e attività inAmazon SWFha bisogno di undominioper correre in (Eseguire)

1. Creazione di un nuovoRegisterDomainRequestobject, fornendo almeno il nome di dominio e il periodo di conservazione dell'esecuzione del flusso di lavoro (questi parametri sono entrambi necessari).
2. Chiama ilAmazon SimpleWorkflowClient.Registra dominiometodo conRegisterDomainRequestoggetto.

3. Cattura ilDomainAlreadyExistsExceptionse il dominio richiesto 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!");
    }
}
```

Elencazione dei domini

Puoi elencareAmazon SWFi domini associati all'account eAWSregione per tipo di registrazione.

1. Creazione di unListDomainsRequestoggetta e specifica lo stato di registrazione dei domini che ti interessano, questo è necessario.
2. Esegui una chiamata aDomini Amazon SimpleWorkflowClient.listcon ilListDomainRequestoggetto. I risultati sono forniti in unDomainInfosoggetto.
3. Esegui una chiamata agetDomainInfossull'oggetto restituito per ottenere un elenco diDomainInfooggetti.
4. Esegui una chiamata agetNamesu ogniDomainInfooggetto per ottenere il suo nome.

Il codice seguente ne è la dimostrazione:

```
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 con l'SDK

LaAWS SDK for JavaViene fornito con esempi di codice che dimostrano molte delle caratteristiche dell'SDK in programmi eseguibili ed eseguibili. Puoi studiarli o modificarli per implementarliAWSsoluzioni che utilizzano laAWS SDK for Java.

Come ottenere i campioni

LaAWS SDK for Javai campioni di codice sono forniti nelcampionidirectory dell'SDK. Se hai scaricato e installato l'SDK utilizzando le informazioni inImpostazione dell'interfacciaAWS SDK for Java, hai già i campioni sul tuo sistema.

È inoltre possibile visualizzare gli ultimi esempi sullaAWS SDK for JavaRepository GitHub, nelsrc/campionidirectory.

Creazione ed esecuzione dei campioni utilizzando la riga di comando

I campioni includonoAntcrea script in modo che tu possa facilmente compilarli ed eseguirli dalla riga di comando. Ogni campione contiene anche un file README in formato HTML che contiene informazioni specifiche per ciascun campione.

> **Note**
>
> Se stai esplorando il codice di esempio su GitHub, fai clic sulRawpulsante nella visualizzazione del codice sorgente durante la visualizzazione del file README.html del campione. In modalità raw, l'HTML verrà visualizzato come previsto nel browser.

Prerequisiti

Prima di eseguire uno qualsiasi deiAWS SDK for Javacampioni, è necessario impostare il tuoAWScredenziali nell'ambiente o con laAWS CLI, come specificato inConfigurazioneAWSCredenziali e regione per lo sviluppo. Gli esempi utilizzano la catena provider di credenziali predefinita quando possibile. Quindi, impostando le tue credenziali in questo modo, puoi

evitare la pratica rischiosa di inserire il tuoAWScredenziali nei file all'interno della directory del codice sorgente (dove potrebbero essere inavvertitamente archiviate e condivise pubblicamente).

Esecuzione degli esempi

1. Passare alla directory contenente il codice del campione. Ad esempio, se ti trovi nella directory principale dellaAWSScarica l'SDK e vuoi eseguire il`AwsConsoleApp`Esempio, digitare:

   ```
   cd samples/AwsConsoleApp
   ```

2. Crea ed esegui il campione con Ant. Il target di compilazione predefinito esegue entrambe le azioni, quindi puoi semplicemente inserire:

   ```
   ant
   ```

Il campione stampa le informazioni sull'output standard, 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 dei campioni utilizzando l'IDE Eclipse

Se utilizzi il pluginAWS Toolkit for Eclipse, è anche possibile avviare un nuovo progetto in Eclipse basato sulAWS SDK for Javaoppure aggiungi l'SDK a un progetto Java esistente.

Prerequisiti

Dopo aver installato il fileAWS Toolkit for Eclipse, ti consigliamo di configurare il Toolkit con le tue credenziali di sicurezza. Tale operazione può essere eseguita in qualsiasi momento scegliendoPreferenzedagliWindowmenu in Eclipse, quindi scegliendo il AWSKit di strumentisezione.

Esecuzione degli esempi

1. Aprire Eclipse
2. Creazione di un nuovoAWSProgetto Java. In Eclipse, sulFilemenu, scegliNovità, quindi fai clic suProgetto. LaNuovo progettosi apre la procedura guidata.
3. Espandere la AWS categoria, quindi scegli AWSProgetto Java.
4. Seleziona Next (Successivo). Viene visualizzata la pagina delle impostazioni del progetto.
5. Immettere un nome nellaNome progetto(Creare snapshot finale?). LaAWS SDK for JavaIl gruppo di campioni visualizza i campioni disponibili nell'SDK, come descritto in precedenza.
6. Selezionare i campioni che si desidera includere nel progetto selezionando ciascuna casella di controllo.
7. Inserisci il tuoAWSCredenziali . Se hai già configurato ilAWS Toolkit for Eclipsecon le tue credenziali, questo viene automaticamente compilato.
8. Scegli Finish (Fine). Il progetto viene creato e aggiunto alProject Explorer.
9. Scegli il campione`.java`file che desideri eseguire. Ad esempio, per laAmazon S3campione, scegli`S3Sample.java`.
10. ScegliereEseguidagliEseguimenu.
11. Fare clic con il pulsante destro delProject Explorer, punto aPercorso di costruzione, quindi scegliAdd library.
12. Scegliere AWSSDK Java, scegliSuccessivo, quindi segui le istruzioni su schermo rimanenti.

Sicurezza per AWS SDK for Java

La sicurezza cloud di Amazon Web Services (AWS) è la priorità più alta. In quanto 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 for Java 1.x
- AWS SDK for Java supporto per TLS
- Identity and Access Management
- Convalida della conformità per questo AWS prodotto o servizio
- Resilienza per questo AWS prodotto o servizio
- Sicurezza dell'infrastruttura per questo AWS prodotto o servizio
- Amazon S3 Migrazione dei client di crittografia

Protezione dei dati in AWS SDK for 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 di mantenere il controllo sui 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.
- Utilizzate SSL/TLS per comunicare con le risorse. AWS
- 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 lavori con questo AWS prodotto o servizio o altri AWS servizi utilizzando la console, l'API o gli SDK. AWS CLI AWS 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 for Java supporto per TLS

Le seguenti informazioni si applicano solo all'implementazione Java SSL (l'implementazione SSL predefinita in). AWS SDK for 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 alcune JVM, 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 - incompatibility-with-aws-sdk - for-java-versions -1-9-5-to-1-10-31/

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 le JVM basate su OpenJDK, è possibile utilizzare la proprietà system. `jdk.tls.client.protocols`

```
java app.jar -Djdk.tls.client.protocols=PROTOCOLS
```

Consulta la documentazione della tua 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 ti 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. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina 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 essere autenticato (aver effettuato l' Utente root dell'account AWS accesso AWS) come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAM Identity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta Signing AWS API request nella IAM User Guide.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conservare le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Identità federata

Come procedura consigliata, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, un provider di identità Web AWS Directory Service, la directory Identity Center o qualsiasi utente che accede Servizi AWS utilizzando credenziali fornite tramite un'origine di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Puoi creare utenti e gruppi in IAM Identity Center oppure puoi connetterti e sincronizzarti con un set di utenti e gruppi nella tua fonte di identità per utilizzarli su tutte le tue applicazioni. Account AWS Per ulteriori informazioni sul Centro identità IAM, consulta Cos'è Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è un'identità interna Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente di IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato IAMAdmins e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in AWS Management Console cambiando ruolo. Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per

ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per ulteriori informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.
- Accesso a più servizi: alcuni Servizi AWS utilizzano le funzionalità di altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso inoltrato (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.

  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire azioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.
- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 e che AWS CLI effettuano richieste API. AWS Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un AWS ruolo a un'istanza EC2 e renderlo disponibile per tutte le sue applicazioni, crei un profilo di istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM

può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'azione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruoli IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente di IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano gli ACL. AWS WAF Per maggiori informazioni sulle ACL, consulta Panoramica delle liste di controllo degli accessi (ACL) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo dei servizi (SCP): le SCP sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità negli account dei membri, inclusa ciascuna. Utente root dell'account AWS Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta 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 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 ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a 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 passare 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 ti 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 tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se AWS supporta queste funzionalità, consultaCome 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, consulta Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente 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. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide introduttive su sicurezza e conformità: queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono i passaggi per l'implementazione di ambienti di base incentrati sulla AWS sicurezza e la conformità.
- Progettazione per la sicurezza e la conformità HIPAA su Amazon Web Services: questo white paper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni idonee all'HIPAA.

> **Note**
>
> Non Servizi AWS tutte sono idonee all'HIPAA. Per ulteriori informazioni, consulta la sezione Riferimenti sui servizi conformi ai requisiti HIPAA.

- AWS Risorse per la per la conformità: questa raccolta di cartelle di lavoro e guide potrebbe essere valida per il tuo settore e la tua località.

- AWS Guide alla conformità dei clienti: comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- Valutazione delle risorse con regole nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida e alle normative del settore.
- AWS Security Hub— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza interno. AWS La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.
- Amazon GuardDuty: Servizio AWS rileva potenziali minacce ai tuoi carichi di lavoro Account AWS, ai contenitori e ai dati monitorando l'ambiente alla ricerca di attività sospette e dannose. GuardDuty può aiutarti a soddisfare vari requisiti di conformità, come lo standard PCI DSS, soddisfacendo i requisiti di rilevamento delle intrusioni imposti da determinati framework di conformità.
- AWS Audit Manager— Ciò Servizio AWS consente di verificare continuamente l' AWS utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

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, puoi 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. O puoi 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 for 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 for 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 utilizza più AWS SDK, è necessario aggiornare ogni SDK separatamente.
2. Esegui la migrazione dei client di crittografia e decrittografia alla versione 2. 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 for 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 for 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 for Java Quindi, ricostruisci il progetto e ridistribuiscilo.

Il completamento di questo passaggio prima di implementare un 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 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 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 for 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 che sono stati 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 `EncryptionOnly`è 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 (CEK) 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 for 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 for Java

Tutti gli artefatti Maven disponibili pubblicamente per il AWS SDK for 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	0xAC107B386692DADD
Type	RSA
Size	4096/4096
Creato	30-06-2016
Scade	2024-10-08
ID utente	AWSSDK e strumenti < @amazon .com> aws-dr-tools
Impronta digitale chiave	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 PAPÀ

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

Questo argomento descrive le modifiche importanti apportate alla AWS SDK for Java Developer Guide nel corso della sua storia.

Questa documentazione è stata creata il: 21 maggio 2024

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 for 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 indica 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 for 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 for Java descrive come generare metriche delle prestazioni delle applicazioni e dell'SDK per. AWS SDK for Java

3 aprile 2017

- Sono stati aggiunti nuovi CloudWatch esempi alla sezione CloudWatch Esempi di utilizzo della AWS SDK for 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 for 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 for 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 for Javaargomento nella sezione Utilizzo della AWS SDK for 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 for 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 for Java, che dimostra come utilizzare i client builder per semplificare la creazione di client. Servizio AWS

  La sezione Esempi di AWS SDK for 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 for 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 for 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 for 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 for 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 AWS il passaggio ai certificati firmati SHA256 per le connessioni SSL e come correggere l'utilizzo di questi certificati negli ambienti Java 1.6 e precedenti, necessari per AWS l'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 configurare le credenziali e la regione per lo sviluppo. AWS

  La discussione sugli esempi di codice è stata spostata in un argomento a parte 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 for 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 for 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 for Java Developer Guide. È inteso come integrazione della cronologia delle note di rilascio.