Guia do desenvolvedor para a versão 1.x

AWS SDK for Java 1.x

AWS SDK for Java 1.x: Guia do desenvolvedor para a versão 1.x

Table of Contents

O AWS SDK for Java 1.x chegou end-of-support em 31 de dezembro de 2025. Recomendamos que você migre para o AWS SDK for Java 2.x para continuar recebendo novos recursos, melhorias de disponibilidade e atualizações de segurança.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.

Guia do desenvolvedor: AWS SDK for Java 1.x

O AWS SDK for Java fornece uma API Java para serviços da AWS. Usando o SDK, você pode compilar facilmente aplicativos Java que funcionem com Amazon S3, Amazon EC2, DynamoDB e muito mais. Sempre adicionamos suporte para novos serviços ao AWS SDK for Java. Para obter uma lista dos serviços compatíveis e as versões da API incluídas com cada versão do SDK, consulte as notas de release da versão com a qual você está trabalhando.

Versão 2 do SDK lançada

Dê uma olhada no novo AWS SDK for Java 2.x em https://github.com/aws/aws-sdk-java-v2/. Ele inclui recursos muito aguardados, como uma maneira de conectar uma implementação HTTP. Para começar a usar, consulte o Guia do desenvolvedor do AWS SDK for Java 2.x.

Documentação e recursos adicionais

Além deste guia, os seguintes recursos online são importantes para desenvolvedores do AWS SDK for Java:

- AWS SDK for Java Referência de API do
- Blog de desenvolvedor Java
- Fóruns de desenvolvedor Java
- GitHub:
  - Fonte da documentação
  - Edições da documentação
  - Fonte do SDK
  - Edições do SDK
  - Exemplos do SDK
  - Canal Gitter
- O Catálogo de exemplos de código da AWS
- @awsforjava (Twitter)
- Notas de release do

Suporte ao IDE do Eclipse

Se desenvolver código usando o IDE do Eclipse, você poderá usar o AWS Toolkit for Eclipse para adicionar o AWS SDK for Java a um projeto do Eclipse existente ou criar um novo projeto do AWS SDK for Java. O toolkit também dá suporte à criação e ao upload de funções Lambda, à inicialização e ao monitoramento de instâncias do Amazon EC2, ao gerenciamento de usuários e grupos de segurança do IAM, a um editor de modelos do AWS CloudFormation e muito mais.

Consulte o Guia do usuário do AWS Toolkit for Eclipse para obter a documentação completa.

Desenvolver aplicativos para Android

Se você for um desenvolvedor para Android, a Amazon Web Services publicará um SDK criado especificamente para desenvolvimento em Android: Amplify Android (o AWS Mobile SDK para Android).

Visualizar o histórico de revisões do SDK

Para visualizar o histórico de versões do AWS SDK for Java, inclusive alterações e serviços compatíveis por versão do SDK, consulte as notas de release do SDK.

Compilar documentação de referência do Java para versões do SDK anteriores

A Referência de API do AWS SDK for Java representa a compilação mais recente da versão 1.x do SDK. Se estiver usando uma compilação anterior da versão 1.x, será possível acessar a documentação de referência do SDK correspondente à versão que está usando.

A maneira mais fácil de compilar a documentação é usar a ferramenta de compilação Maven do Apache. Faça download e instale primeiro o Maven se você ainda não o tiver no sistema e use as instruções a seguir para compilar a documentação de referência.

1. Localize e selecione a versão do SDK que você está usando na página versões do repositório do SDK no GitHub.
2. Escolha o link `zip` (a maioria das plataformas, inclusive Windows) ou `tar.gz` (Linux, macOS ou Unix) para fazer download do SDK no computador.

3. Descompacte o arquivo em um diretório local.
4. Na linha de comando, navegue até o diretório onde você descompactou o arquivo e digite o seguinte.

```
mvn javadoc:javadoc
```

5. Depois da conclusão da compilação, você encontrará a documentação HTML gerada no diretório `aws-java-sdk/target/site/apidocs/`.

Conceitos básicos

Esta seção fornece informações sobre como instalar, configurar e usar o AWS SDK for Java.

Tópicos

Configuração básica para trabalhar com os Serviços da AWS

Visão geral

Para desenvolver com êxito aplicativos que acessem os Serviços da AWS usando o AWS SDK for Java, as seguintes condições são necessárias:

- Você deve conseguir entrar no portal de acesso da AWS disponível em Centro de Identidade do AWS IAM.
- As permissões do perfil do IAM configuradas no SDK devem permitir o acesso aos Serviços da AWS que seu aplicativo exige. As permissões associadas à política gerenciada PowerUserAccess da AWS são suficientes para a maioria das necessidades de desenvolvimento.
- Um ambiente de desenvolvimento com os seguintes elementos:
  - Arquivos de configuração compartilhados que são configurados da seguinte forma:
    - O arquivo `config` contém um perfil padrão que especifica uma Região da AWS.
    - O arquivo `credentials` contém credenciais temporárias como parte de um perfil padrão.
  - Uma instalação do Java adequada.
  - Uma ferramenta de automação de compilação, como Maven ou Gradle.
  - Um editor de texto para trabalhar com código.
  - (Opcional, mas recomendado) Um IDE (ambiente de desenvolvimento integrado), como IntelliJ IDEA, Eclipse ou NetBeans.

Ao usar um IDE, você também pode integrar AWS Toolkits para trabalhar com mais facilidade com Serviços da AWS. O AWS Toolkit para IntelliJ e o AWS Toolkit for Eclipse são dois kits de ferramentas que você pode usar para desenvolvimento em Java.

> **Important**
>
> As instruções nesta seção de configuração pressupõem que você ou a organização usam o IAM Identity Center. Se sua organização usa um provedor de identidades externo que funciona independentemente do IAM Identity Center, descubra como você pode obter credenciais temporárias para o SDK para Java usar. Siga estas instruções para adicionar credenciais temporárias ao arquivo `~/.aws/credentials`.
> Se seu provedor de identidade adicionar credenciais temporárias automaticamente ao arquivo `~/.aws/credentials`, certifique-se de que o nome do perfil seja `[default]` para que você não precise fornecer um nome de perfil ao SDK ou à AWS CLI.

Capacidade de efetuar login no portal de acesso da AWS

O portal de acesso da AWS é o local da web em que você faz login manualmente no IAM Identity Center. O formato da URL é `d-xxxxxxxxxx.awsapps.com/start` ou *`your_subdomain`*`.awsapps.com/start`.

Se você não estiver familiarizado com o portal de acesso da AWS, siga as orientações para acesso à conta na Etapa 1 do tópico de autenticação do IAM Identity Center no Guia de referência de SDKs e ferramentas da AWS. Não siga a Etapa 2 porque o AWS SDK for Java 1.x não oferece suporte à atualização automática de tokens e à recuperação automática de credenciais temporárias para o SDK que a Etapa 2 descreve.

Definir arquivos de configuração compartilhados

Os arquivos de configuração compartilhados residem na sua estação de trabalho de desenvolvimento e contêm configurações básicas usadas por todos os SDKs da AWS e pela AWS Command Line Interface (CLI). Os arquivos de configuração compartilhados podem conter várias configurações, mas essas instruções definem os elementos básicos necessários para trabalhar com o SDK.

Configurar o arquivo compartilhado **`config`**

O exemplo a seguir mostra o conteúdo de um arquivo compartilhado `config`.

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

Para fins de desenvolvimento, use a Região da AWS mais próxima de onde você planeja executar seu código. Para obter uma lista dos códigos de região a serem usados no arquivo `config`, consulte o guia Referência geral da Amazon Web Services. A configuração `json` do formato de saída é um dos vários valores possíveis.

Siga as orientações nesta seção para criar o arquivo `config`.

Configurar credenciais temporárias para o SDK

Depois de ter acesso a uma Conta da AWS e a um perfil do IAM por meio do portal de acesso da AWS, configure seu ambiente de desenvolvimento com credenciais temporárias para o SDK acessar.

Etapas para configurar um arquivo local **`credentials`** com credenciais temporárias

1. Crie um arquivo compartilhado `credentials`.
2. No arquivo de `credentials`, cole o texto do espaço reservado a seguir até colar as credenciais temporárias de trabalho.

   ```
   [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. Salve o arquivo. Agora, o arquivo `~/.aws/credentials` deve existir em seu sistema de desenvolvimento local. Esse arquivo contém o perfil [padrão] que o SDK para Java usa se um perfil nomeado específico não for especificado.
4. Faça login no portal de acesso da AWS.
5. Siga estas instruções no título Atualizar credencial manual para copiar credenciais do perfil do IAM do portal de acesso AWS.

a. Na etapa 4 das instruções vinculadas, escolha o nome do perfil do IAM que concede acesso para suas necessidades de desenvolvimento. Esse perfil normalmente tem um nome como PowerUserAccess ou Developer.

b. Na etapa 7, selecione a opção Adicionar manualmente um perfil ao seu arquivo de credenciais da AWS e copie o conteúdo.

6. Cole as credenciais copiadas em seu arquivo local `credentials` e remova qualquer nome de perfil que tenha sido colado. Seu arquivo deve se parecer com o seguinte:

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

7. Salve o arquivo `credentials`

O SDK para Java acessará essas credenciais temporárias quando cria um cliente de serviço e as usa para cada solicitação. As configurações do perfil do IAM escolhidas na etapa 5a determinam por quanto tempo as credenciais temporárias são válidas. A duração máxima é de doze horas.

Depois que as credenciais temporárias expirarem, repita as etapas de 4 a 7.

Instalar um ambiente de desenvolvimento Java

O AWS SDK for Java V1 requer um Java 7 JDK ou mais recente e todas as versões do Java LTS (suporte de longo prazo) do JDK são compatíveis. Se você usa a versão 1.12.767 ou anterior do SDK, pode usar o Java 7, mas se usa a versão 1.12.768 ou mais recente do SDK, o Java 8 é necessário. O repositório central do Maven lista a versão mais recente do SDK para Java.

O AWS SDK for Java funciona com o Oracle Java SE Development Kit e com distribuições do Open Java Development Kit (OpenJDK), como Amazon Corretto, Red Hat OpenJDK e Adoptium.

Maneiras de obter o AWS SDK for Java

Pré-requisitos

Para usar o AWS SDK for Java, você deve ter:

- Você deve conseguir entrar no portal de acesso da AWS disponível em Centro de Identidade do AWS IAM.
- Uma instalação do Java adequada.
- Credenciais temporárias configuradas em seu arquivo compartilhado `credentials` local.

Consulte o tópico the section called “Configuração básica” para obter instruções sobre como se preparar para usar o SDK para Java.

Usar uma ferramenta de compilação para gerenciar dependências do SDK para Java (recomendado)

Recomendamos usar o Apache Maven ou o Gradle com seu projeto para acessar as dependências necessárias do SDK para Java. Esta seção descreve como usar essas ferramentas.

Baixar e extrair o SDK (não recomendado)

Recomendamos que você use uma ferramenta de compilação para acessar o SDK do seu projeto. No entanto, você pode baixar um jar pré-compilado da versão mais recente do SDK.

> **Note**
>
> Para obter informações sobre como fazer download e compilar versões anteriores do SDK, consulte Instalar versões anteriores do SDK.

1. Baixe o SDK em https://sdk-for-java.amazonwebservices.com/latest/aws-java-sdk.zip.
2. Depois de fazer download do SDK, extraia o conteúdo em um diretório local.

O SDK contém os seguintes diretórios:

- `documentation`: contém a documentação da API (também disponível na web: Referência de API do AWS SDK for Java).
- `lib`: contém os arquivos `.jar` do SDK.
- `samples`: contém código de exemplo funcional que demonstra como usar o SDK.
- `third-party/lib`: contém bibliotecas de terceiros usadas pelo SDK, como registro em log comum do Apache, AspectJ e a estrutura do Spring.

Para usar o SDK, adicione o caminho completo dos diretórios `lib` e `third-party` às dependências no arquivo de compilação e os adicione ao `CLASSPATH` Java para executar o código.

Compilar versões anteriores do SDK a partir da fonte (não recomendado)

Somente a versão mais recente do SDK completo é fornecida na forma pré-compilada como jar disponível para download. No entanto, você pode compilar uma versão anterior do SDK usando o Apache Maven (código-fonte aberto). O Maven vai fazer download de todas as dependências necessárias, compilar e instalar o SDK em uma única etapa. Visite http://maven.apache.org/ para obter instruções de instalação e mais informações.

1. Acesse a página GitHub do SDK em: AWS SDK for Java (GitHub).
2. Escolha a tag correspondente ao número de versão do SDK desejada. Por exemplo, `1.6.10`.
3. Clique no botão Download ZIP para fazer download da versão do SDK selecionada por você.
4. Descompacte o arquivo em um diretório no sistema de desenvolvimento. Em muitos sistemas, você pode usar o gerenciador de arquivos gráficos para fazer isso ou usar o utilitário `unzip` em uma janela de terminal.
5. Em uma janela de terminal, navegue até o diretório em que você descompactou o código-fonte do SDK.
6. Compile e instale o SDK com o seguinte comando (Maven obrigatório):

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

   O arquivo `.jar` resultante foi compilado no diretório `target`.
7. (Opcional) Compile a documentação de referência da API usando o seguinte comando:

   ```
   mvn javadoc:javadoc
   ```

   A documentação foi compilada no diretório `target/site/apidocs/`.

Usar ferramentas de compilação

O uso de ferramentas de compilação ajuda a gerenciar o desenvolvimento de projetos Java. Várias ferramentas de compilação estão disponíveis, mas mostramos como começar a usar duas ferramentas de compilação populares: Maven e Gradle. Este tópico mostra como usar essas

ferramentas de compilação para gerenciar as dependências do SDK para Java necessárias para seus projetos.

Tópicos

Usar o SDK com o Apache Maven

Você pode usar o Apache Maven para configurar e compilar projetos do AWS SDK for Java ou compilar o próprio SDK.

> **Note**
>
> Você deve ter o Maven instalado para usar a diretriz deste tópico. Se ele ainda não estiver instalado, visite http://maven.apache.org/ para fazer download e instalá-lo.

Criar um novo pacote do Maven

Para criar um pacote do Maven básico, abra uma janela de terminal (linha de comando) e execute:

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

Substitua org.example.basicapp pelo namespace do pacote completo do aplicativo e myapp pelo nome do projeto (este será o nome do diretório do projeto).

Por padrão, cria um modelo de projeto para você usando o arquétipo quickstart, que é um bom ponto de partida para muitos projetos. Existem mais arquétipos disponíveis; visite a página Arquétipos do Maven para obter uma lista de arquétipos empacotados. Você pode escolher um determinado arquétipo a ser usado adicionando o argumento `-DarchetypeArtifactId` ao comando `archetype:generate`. Por exemplo:

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

```
  -DgroupId=org.example.webapp \
  -DartifactId=mywebapp
```

> **Note**
>
> Muito mais informações sobre como criar e configurar projetos são fornecidas no Guia de conceitos básicos do Maven.

Configurar o SDK como uma dependência do Maven

Para usar o AWS SDK for Java no projeto, será necessário declará-lo como uma dependência no arquivo `pom.xml` do projeto. Desde a versão 1.9.0, você pode importar componentes individuais ou todo o SDK.

Especificar módulos de SDK individuais

Para selecionar módulos de SDK individuais, use a lista de materiais (BOM) do AWS SDK for Java para Maven, o que garantirá que os módulos especificados usem a mesma versão do SDK e sejam compatíveis entre si.

Para usar a BOM, adicione uma seção `<dependencyManagement>` ao arquivo `pom.xml` do aplicativo, adicionando `aws-java-sdk-bom` como uma dependência e especificando a versão do SDK que você deseja usar:

```
<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>
```

Para visualizar a versão mais recente da BOM do AWS SDK for Java disponível no Maven Central, visite: https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-bom. Também é possível usar essa página para ver quais módulos (dependências) são gerenciados pela BOM que é possível incluir na seção `<dependencies>` do arquivo `pom.xml` do projeto.

Agora você pode selecionar módulos individuais do SDK usados no aplicativo. Como já declarou a versão do SDK na BOM, você não precisa especificar o número da versão de cada 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>
```

Também é possível consultar o Catálogo de exemplos de código da AWS para saber quais dependências devem ser usadas para determinado AWS service (Serviço da AWS). Consulte o arquivo POM em um exemplo de serviço específico. Por exemplo, se você estiver interessado nas dependências do serviço do AWS S3, consulte o exemplo completo no GitHub. (Examine o pom em / java/example_code/s3).

Importar todos os módulos do SDK

Se você quiser extrair todo o SDK como uma dependência, não use o método da BOM. Basta declará-lo no `pom.xml` da seguinte forma:

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

Compilar o projeto

Depois de configurar o projeto, será possível criá-lo usando o comando `package` do Maven:

```
mvn package
```

Isso criará o arquivo `0jar` no diretório `target`.

Compilar o SDK com o Maven

Você pode usar o Apache Maven para compilar o SDK pela origem. Para isso, faça download do código do SDK no GitHub, descompacte-o localmente e execute o seguinte comando do Maven:

```
mvn clean install
```

Usar o SDK com o Gradle

Para gerenciar dependências do SDK para seu projeto Gradle, importe a BOM Maven do AWS SDK for Java para o arquivo `build.gradle` do aplicativo.

> **Note**
>
> Nos exemplos a seguir, substitua *`1.12.529`* no arquivo de compilação por uma versão válida do AWS SDK for Java. Encontre a versão mais recente no repositório central do Maven.

Configuração do projeto para Gradle 4.6 ou superior

Desde o Gradle 4.6, é possível usar o recurso de suporte de POM aprimorado do Gradle para a importação de arquivos de lista de materiais (BOM) declarando uma dependência em uma BOM.

1. Se você estiver usando o Gradle 5.0 ou posterior, pule para a etapa 2. Caso contrário, habilite o recurso IMPROVED_POM_SUPPORT no arquivo `settings.gradle`.

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

2. Adicione a BOM à seção dependências do arquivo `build.gradle` do aplicativo.

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

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

3. Especifique os módulos do SDK a serem usados na seção dependencies. Por exemplo, o seguinte inclui uma dependência para o Amazon Simple Storage Service (Amazon S3).

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

O Gradle resolve automaticamente a versão correta das dependências do SDK usando as informações da BOM.

Veja a seguir um exemplo de um arquivo `build.gradle` completo que inclui uma dependência para o 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**
>
> No exemplo anterior, substitua a dependência para o Amazon S3 pelas dependências dos serviços da AWS que você usará no seu projeto. Os módulos (dependências) que são gerenciados pela BOM do AWS SDK for Java estão listados no repositório central do Maven.

Configuração do projeto para versões do Gradle anteriores à 4.6

As versões do Gradle anteriores à 4.6 não possuem suporte nativo a BOM. Para gerenciar dependências do AWS SDK for Java para o seu projeto, use o plug-in de gerenciamento de dependências do Spring para Gradle para importar a BOM Maven para o SDK.

1. Adicione o plug-in de gerenciamento de dependências ao arquivo `build.gradle` do aplicativo.

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

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

2. Adicione a BOM à seção dependencyManagement do arquivo.

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

3. Especifique os módulos do SDK que você usará na seção dependencies Por exemplo, o seguinte inclui uma dependência para o Amazon S3.

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

O Gradle resolve automaticamente a versão correta das dependências do SDK usando as informações da BOM.

Veja a seguir um exemplo de um arquivo `build.gradle` completo que inclui uma dependência para o 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**
>
> No exemplo anterior, substitua a dependência para o Amazon S3 pelas dependências do serviço da AWS que você usará no seu projeto. Os módulos (dependências) que são gerenciados pela BOM do AWS SDK for Java estão listados no repositório central do Maven.

Para obter mais informações sobre como especificar dependências do SDK usando a BOM, consulte Usar o SDK com o Apache Maven.

Configurar as credenciais temporárias da AWS e a Região da AWS para desenvolvimento

Para se conectar a qualquer um dos serviços compatíveis com o AWS SDK for Java, você deve fornecer as credenciais temporárias da AWS. Os SDKs e as CLIs da AWS usam cadeias de fornecedores para procurar credenciais temporárias da AWS em vários lugares diferentes, inclusive variáveis de sistema/ambiente do usuário e arquivos de configuração da AWS locais.

Este tópico fornece informações básicas sobre como configurar as credenciais temporárias da AWS para desenvolvimento de aplicativos locais usando o AWS SDK for Java. Se for necessário configurar credenciais a serem usadas dentro de uma instância do EC2 ou se estiver usando o IDE do Eclipse para desenvolvimento, consulte os seguintes tópicos:

- Ao usar uma instância do EC2, crie um perfil do IAM e dê à instância do EC2 acesso a essa função conforme mostrado em Usar perfis do IAM para conceder acesso a recursos da AWS no Amazon EC2.
- Configure as credenciais da AWS no Eclipse usando o AWS Toolkit for Eclipse. Consulte Configurar credenciais da AWS no Guia do usuário do AWS Toolkit for Eclipse para obter mais informações.

Configurar credenciais temporárias

É possível configurar credenciais temporárias para o AWS SDK for Java de várias maneiras, mas aqui estão as abordagens recomendadas:

- Defina credenciais temporárias no arquivo de perfil de credenciais da AWS no sistema local, localizado em:
  - `~/.aws/credentials` no Linux, macOS ou Unix
  - `C:\Users\USERNAME\.aws\credentials` no Windows

  Consulte the section called “Configurar credenciais temporárias para o SDK” neste guia para obter instruções sobre como obter suas credenciais temporárias.
- Defina as variáveis de ambiente `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` e `AWS_SESSION_TOKEN`.

  Para definir essas variáveis no Linux, macOS ou Unix, use :

```
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
```

Para definir essas variáveis no Windows, use :

```
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
```

- Para uma instância do EC2, especifique um perfil do IAM e dê à instância do EC2 acesso a essa função. Consulte Perfis do IAM para Amazon EC2 no Guia do usuário do Amazon EC2 para instâncias do Linux para uma discussão detalhada sobre como isso funciona.

Depois que você tiver definido as credenciais temporárias da AWS usando um desses métodos, elas serão carregadas automaticamente pelo AWS SDK for Java usando-se a cadeia de fornecedores de credencial padrão. Para obter mais informações sobre como trabalhar com credenciais da AWSnos aplicativos Java, consulte Trabalhar com credenciais da AWS.

Atualizar credenciais do IMDS

O AWS SDK for Java permite atualizar credenciais do IMDS em segundo plano a cada minuto, independentemente do tempo de expiração da credencial. Isso permite que você atualize as credenciais com mais frequência e reduz a chance de que não atingir o IMDS afete a disponibilidade percebida da AWS.

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

```
12. // This is new: When you are done with the credentials provider, you must close it
to release the background thread.
13. credentials.close();
```

Defina a Região da AWS

Você deve definir uma Região da AWS padrão que será usada para acessar serviços da AWS com o AWS SDK for Java. Tendo em vista o melhor desempenho da rede, você deve escolher uma região geograficamente próxima de você (ou dos clientes). Para obter uma lista de regiões para cada serviço, consulte Regiões e endpoints na Referência geral da Amazon Web Services.

> **Note**
>
> Se você não selecionar uma região, us-east-1 será usada por padrão.

Você pode usar técnicas semelhantes para à definição de credenciais para configurar a região da AWS padrão:

- Defina a Região da AWS no arquivo de configuração da AWS no sistema local, localizado em:
  - ~/.aws/config no Linux, macOS ou Unix
  - C:\Users\USERNAME\.aws\config no Windows

  Esse arquivo deve conter linhas no seguinte formato:

  +

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

  +

  Substitua a Região da AWS desejada (por exemplo, “us-west-1”) para your_aws_region.
- Defina a variável de ambiente `AWS_REGION`.

  No Linux, macOS ou Unix, use :

  ```
  export AWS_REGION=your_aws_region
  ```

No Windows, use :

```
set AWS_REGION=your_aws_region
```

Em que your_aws_region é o nome da Região da AWS desejado.

Usando o AWS SDK for Java

Esta seção fornece informações gerais importantes sobre programação com o AWS SDK for Java que se aplicam a todos os serviços que você pode usar com o SDK.

Para obter informações e exemplos de programação específicos do serviço (para Amazon EC2, Amazon S3 Amazon SWF, etc.), consulte Exemplos de AWS SDK for Java código.

Tópicos

Melhores práticas para AWS desenvolvimento com o AWS SDK for Java

As práticas recomendadas a seguir podem ajudá-lo a evitar problemas ou problemas ao desenvolver AWS aplicativos com AWS SDK for Java o. Organizamos as melhores práticas por serviço.

S3

Evite ResetExceptions

Ao fazer upload de objetos usando fluxos (Amazon S3 por meio de um `AmazonS3` cliente ou`TransferManager`), você pode encontrar problemas de conectividade de rede ou de tempo limite. Por padrão, as AWS SDK for Java tentativas de repetir as transferências falharam marcando

o fluxo de entrada antes do início de uma transferência e, em seguida, redefinindo-o antes de tentar novamente.

Se o stream não suportar a marcação e a redefinição, o SDK lançará um ResetExceptionquando houver falhas transitórias e as novas tentativas forem ativadas.

Melhor prática

Recomendamos usar fluxos que deem suporte a operações de marcar e redefinir.

A maneira mais confiável de evitar a ResetExceptioné fornecer dados usando um arquivo ou FileInputStream, que eles AWS SDK for Java possam manipular sem serem limitados por limites de marcação e redefinição.

Se o stream não for um FileInputStream, mas oferecer suporte à marcação e à redefinição, você poderá definir o limite de marcas usando o `setReadLimit` método de RequestClientOptions. O valor padrão é 128 KB. Definir o valor limite de leitura como um byte maior que o tamanho do fluxo evitará de forma confiável a. ResetException

Por exemplo, se o tamanho esperado máximo de um fluxo for 100.000 bytes, defina o limite de leitura como 100.001 (100.000 + 1) bytes. A marca e a redefinição sempre funcionarão para 100.000 bytes ou menos. Lembre-se de que isso pode fazer alguns fluxos armazenarem em buffer esse número de bytes na memória.

Criar clientes de serviço

Para fazer solicitações Amazon Web Services, primeiro você cria um objeto de cliente de serviço. A maneira recomendada é usar o compilador de cliente do serviço.

Cada um AWS service (Serviço da AWS) tem uma interface de serviço com métodos para cada ação na API de serviço. Por exemplo, a interface de serviço do DynamoDB é chamada. AmazonDynamoDBClient Cada interface de serviço tem um compilador de cliente correspondente que você pode usar para construir uma implementação da interface de serviço. A classe do construtor de clientes DynamoDB se chama AmazonDynamoDBClientBuilder.

Obter um compilador de cliente

Para obter uma instância do compilador de cliente, use o método de fábrica estático `standard`, conforme mostrado no exemplo a seguir.

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

Assim que tiver um compilador, será possível personalizar as propriedades do cliente usando muitos setters fluentes na API do compilador. Por exemplo, você pode definir uma região e um provedor de credenciais personalizados da maneira a seguir.

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

> **Note**
>
> Os métodos `withXXX` fluentes retornam o objeto `builder`, de maneira que você possa vincular as chamadas ao método por comodidade e um código mais legível. Depois de configurar as propriedades desejadas, você poderá chamar o método `build` para criar o cliente. Assim que for criado, um cliente será imutável, e todas as chamadas para `setRegion` ou `setEndpoint` falharão.

Um compilador pode criar vários clientes com a mesma configuração. Quando você estiver escrevendo o aplicativo, lembre-se de que o compilador será mutável e não será thread-safe.

O código a seguir usa o compilador como uma fábrica para instâncias de cliente.

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

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

O construtor também expõe configuradores fluentes para ClientConfiguratione RequestMetricCollector, e uma lista personalizada de 2. RequestHandler

Este é um exemplo completo que substitui todas as propriedades configuráveis.

```
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();
```

Criar clientes async

AWS SDK for Java Tem clientes assíncronos (ou assíncronos) para cada serviço (exceto para Amazon S3) e um construtor de clientes assíncronos correspondente para cada serviço.

Para criar um cliente assíncrono do DynamoDB com o padrão ExecutorService

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

Além das opções de configuração suportadas pelo construtor de clientes síncronos (ou sincronizados), o cliente assíncrono permite que você defina um personalizado ExecutorFactorypara alterar o `ExecutorService` que o cliente assíncrono usa. `ExecutorFactory`é uma interface funcional, portanto, interopera com expressões lambda e referências de métodos do Java 8.

Para criar um cliente async com um executor personalizado

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

Usando DefaultClient

Os compiladores de cliente sync e async têm outro método de fábrica chamado `defaultClient`. Esse método cria um cliente de serviço com a configuração padrão usando a cadeia de fornecedores padrão para carregar credenciais e a Região da AWS. Se as credenciais ou a região não puderem ser determinadas pelo ambiente no qual o aplicativo estiver em execução, a chamada a `defaultClient` falhará. Consulte Trabalhando com AWS credenciais e Região da AWS seleção para obter mais informações sobre como as credenciais e a região são determinadas.

Para criar um cliente de serviço padrão

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

Ciclo de vida do cliente

Os clientes de serviço no SDK são thread-safe e, tendo em vista o melhor desempenho, você deve tratá-los como objetos de longa duração. Cada cliente tem o próprio recurso do grupo de conexões. Desligue explicitamente clientes quando eles não são mais necessários para evitar vazamentos de recursos.

Para desligar explicitamente um cliente, chame o método `shutdown`. Depois da chamada de `shutdown`, todos os recursos de cliente serão lançados, e o cliente será inutilizável.

Para desligar um cliente

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

Forneça credenciais temporárias ao AWS SDK for Java

Para fazer solicitações Amazon Web Services, você deve fornecer credenciais AWS temporárias para AWS SDK for Java que o use ao chamar os serviços. Isso pode ser feito das seguintes maneiras:

- Use a cadeia de fornecedores de credenciais padrão (recomendado).
- Use um fornecedor de credenciais específico ou uma cadeia de fornecedores (ou crie a própria).
- Forneça você mesmo as credenciais temporárias em código.

Usar a cadeia de fornecedores de credenciais padrão

Quando você inicializa um novo cliente de serviço sem fornecer nenhum argumento, ele AWS SDK for Java tenta encontrar credenciais temporárias usando a cadeia de fornecedores de credenciais padrão implementada pela classe Default. AWSCredentials ProviderChain A cadeia de fornecedores de credenciais padrão procura credenciais nesta ordem:

1. Variáveis de ambiente: `AWS_ACCESS_KEY_ID`, `AWS_SECRET_KEY` ou `AWS_SECRET_ACCESS_KEY` e `AWS_SESSION_TOKEN`. O AWS SDK for Java usa a EnvironmentVariableCredentialsProviderclasse para carregar essas credenciais.
2. Propriedades do sistema Java: `aws.accessKeyId`, `aws.secretKey` (mas não `aws.secretAccessKey`) e `aws.sessionToken`. O AWS SDK for Java usa o SystemPropertiesCredentialsProviderpara carregar essas credenciais.
3. Credenciais de token de identidade da Web do ambiente ou contêiner.
4. O arquivo de perfis de credenciais padrão - normalmente localizado em `~/.aws/credentials` (a localização pode variar de acordo com a plataforma) e compartilhado por muitos dos AWS SDKs e pelos AWS CLI. O AWS SDK for Java usa o ProfileCredentialsProviderpara carregar essas credenciais.

   Você pode criar um arquivo de credenciais usando o `aws configure` comando fornecido pelo AWS CLI ou pode criá-lo editando o arquivo com um editor de texto. Para obter mais informações sobre o formato do arquivo de credenciais, consulte Formato do arquivo de credenciais da AWS.
5. Credenciais de contêiner do Amazon ECS: carregadas pelo Amazon ECS se a variável de ambiente `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` estiver definida. O AWS SDK for Java usa o ContainerCredentialsProviderpara carregar essas credenciais. É possível especificar o endereço IP para esse valor.
6. Credenciais de perfil de instância — usadas em EC2 instâncias e fornecidas por meio do serviço de Amazon EC2 metadados. O AWS SDK for Java usa o InstanceProfileCredentialsProviderpara carregar essas credenciais. É possível especificar o endereço IP para esse valor.

> **Note**
>
> As credenciais de perfil de instância serão usadas somente se `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` não estiver definido. Consulte EC2ContainerCredentialsProviderWrapper para obter mais informações.

Configurar credenciais temporárias

Para poder usar credenciais AWS temporárias, elas devem ser definidas em pelo menos um dos locais anteriores. Para obter informações sobre como configurar credenciais, consulte os seguintes tópicos:

- Para especificar credenciais no ambiente ou no arquivo de perfis de credencial padrão, consulte the section called “Configurar credenciais temporárias”.
- Para definir as propriedades de sistema do Java, consulte o tutorial Propriedades do sistema no site Tutoriais do Java oficial.
- Para configurar e usar as credenciais do perfil da instância com suas EC2 instâncias, consulte Como usar funções do IAM para conceder acesso a AWS recursos em Amazon EC2.

Configurar um perfil de credenciais alternativo

O AWS SDK for Java usa o perfil padrão por padrão, mas há maneiras de personalizar qual perfil é originado do arquivo de credenciais.

Você pode usar a variável AWS de ambiente Profile para alterar o perfil carregado pelo SDK.

Por exemplo, no Linux, no macOS ou no Unix, você executaria o comando a seguir a fim de alterar o perfil para myProfile.

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

No Windows, você usaria o seguinte.

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

A configuração da variável de `AWS_PROFILE` ambiente afeta o carregamento de credenciais de todas as ferramentas AWS SDKs e ferramentas oficialmente suportadas (incluindo a AWS CLI e a AWS Tools for Windows PowerShell). Para alterar somente o perfil de um aplicativo Java, você pode usar a propriedade do sistema `aws.profile` em seu lugar.

> **Note**
>
> A variável de ambiente tem precedência sobre a propriedade do sistema.

Configurar um local de arquivo de credenciais alternativo

O AWS SDK for Java carrega credenciais AWS temporárias automaticamente do local padrão do arquivo de credenciais. No entanto, você também pode especificar o local configurando a variável

de ambiente `AWS_CREDENTIAL_PROFILES_FILE` com o caminho completo para o arquivo de credenciais.

Você pode usar esse recurso para alterar temporariamente o local em que ele AWS SDK for Java procura seu arquivo de credenciais (por exemplo, definindo essa variável com a linha de comando). Ou você pode definir a variável de ambiente no ambiente de usuário ou sistema a fim de alterá-la para o usuário ou o sistema.

Para substituir o local do arquivo de credenciais padrão

- Defina a variável de `AWS_CREDENTIAL_PROFILES_FILE` ambiente para o local do seu arquivo de AWS credenciais.
  - No Linux, macOS ou Unix, use:

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

  - No Windows, use:

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

Formato do arquivo **`Credentials`**

Seguindo as instruções na Configuração básica deste guia, seu arquivo de credenciais deve ter o seguinte formato básico.

```
[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>
```

O nome do perfil é especificado entre colchetes (por exemplo, `[default]`), seguido dos campos configuráveis nesse perfil como pares de chave/valor. Você pode ter vários perfis no arquivo `credentials`, que podem ser adicionados ou editados usando-se `aws configure --profile` *`PROFILE_NAME`* para selecionar o perfil a ser configurado.

Você pode especificar campos adicionais, como `metadata_service_timeout` e `metadata_service_num_attempts`. Eles não são configuráveis com a CLI. Você deverá editar o arquivo manualmente se quiser usá-los. Para obter mais informações sobre o arquivo de configuração e seus campos disponíveis, consulte Configurando o AWS Command Line Interface no Guia do AWS Command Line Interface Usuário.

Carregar credenciais

Depois que definir as credenciais temporárias, o SDK as carrega usando a cadeia de fornecedores de credenciais padrão.

Para fazer isso, você instancia um AWS service (Serviço da AWS) cliente sem fornecer explicitamente as credenciais ao construtor, da seguinte maneira.

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

Especificar um fornecedor de credenciais ou uma cadeia de fornecedores

Você pode especificar um fornecedor de credenciais diferente do fornecedor de credenciais padrão usando o compilador de cliente.

Você fornece uma instância de um provedor de credenciais ou cadeia de fornecedores para um criador de clientes que usa uma interface de AWSCredentialsprovedor como entrada. O exemplo a seguir mostra como usar credenciais de ambiente mais especificamente.

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

Para ver a lista completa de provedores AWS SDK for Java de credenciais e cadeias de provedores fornecidos, consulte Todas as classes de implementação conhecidas no AWSCredentials provedor.

> **Note**
>
> Você pode usar essa técnica para fornecer provedores de credenciais ou cadeias de provedores que você cria usando seu próprio provedor de credenciais que

implementa a `AWSCredentialsProvider` interface ou subclassificando a classe. AWSCredentialsProviderChain

Especificar explicitamente credenciais temporárias

Se a cadeia de credenciais padrão ou um fornecedor personalizado ou específico ou a cadeia de fornecedores não funcionar para o código, será possível definir credenciais fornecidas explicitamente. Se você recuperou credenciais temporárias usando AWS STS, use esse método para especificar as credenciais de acesso. AWS

1. Instancie a BasicSessionCredentialsclasse e forneça a ela a chave de AWS acesso, a chave AWS secreta e o token de AWS sessão que o SDK usará para a conexão.
2. Crie um AWSStaticCredentialsProvidercom o `AWSCredentials` objeto.
3. Configure o compilador de cliente com o `AWSStaticCredentialsProvider` e compilar o cliente.

Veja um exemplo do a seguir:

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

Mais informações

- Inscreva-se AWS e crie um usuário do IAM
- Configurar AWS credenciais e região para desenvolvimento
- Usando funções do IAM para conceder acesso a AWS recursos em Amazon EC2

Região da AWS Seleção

As regiões permitem que você acesse AWS serviços que residem fisicamente em uma área geográfica específica. Isso pode ser útil para redundância e para manter os dados e os aplicativos em execução próximo ao lugar onde você e os usuários os acessarão.

Verificar disponibilidade do serviço em uma região

Para ver se um determinado AWS service (Serviço da AWS) está disponível em uma região, use o `isServiceSupported` método na região que você gostaria de usar.

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

Consulte a documentação da classe Regions das regiões que você pode especificar e usar o prefixo de endpoint do serviço a ser consultado. O prefixo de endpoint de cada serviço é definido na interface de serviço. Por exemplo, o prefixo do DynamoDB endpoint é definido em AmazonDynamoDB.

Escolher uma região

A partir da versão 1.4 do AWS SDK for Java, você pode especificar um nome de região e o SDK escolherá automaticamente um endpoint apropriado para você. Para escolher o endpoint por conta própria, consulte Escolher um endpoint específico.

Para definir explicitamente uma região, recomendamos usar o enum Regions. Esta é uma enumeração de todas as regiões disponíveis publicamente. Para criar um cliente com uma região do enum, use o código a seguir.

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

Se a região que estiver tentando usar não estiver no enum `Regions`, será possível definir a região usando uma string que represente o nome da região.

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

> **Note**
>
> Depois que você compilar um cliente com o compilador, ele será imutável, e a região não poderá ser alterada. Se você estiver trabalhando com vários Regiões da AWS para o mesmo serviço, deverá criar vários clientes — um por região.

Escolher um endpoint específico

Cada AWS cliente pode ser configurado para usar um endpoint específico em uma região chamando o `withEndpointConfiguration` método ao criar o cliente.

Por exemplo, para configurar o Amazon S3 cliente para usar a região da Europa (Irlanda), use o código a seguir.

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

Consulte Regiões e endpoints para ver a lista atual de regiões e seus endpoints correspondentes para todos os AWS serviços.

Determinar automaticamente a região pelo ambiente

> **Important**
>
> Esta seção se aplica somente ao usar um construtor de clientes para acessar AWS serviços. AWS os clientes criados usando o construtor do cliente não determinarão automaticamente a região do ambiente e, em vez disso, usarão a região padrão do SDK ()USEast1.

Ao executar no Lambda Amazon EC2 ou no Lambda, talvez você queira configurar os clientes para usar a mesma região em que seu código está sendo executado. Isso desvincula o código do ambiente no qual está em execução e facilita ainda mais a implantação do aplicativo em várias regiões tendo em vista menos latência ou redundância.

Você deve usar compiladores de cliente para que o SDK detecte automaticamente a região onde o código está sendo executado.

Para usar a cadeia de credential/region fornecedores padrão para determinar a região a partir do ambiente, use o `defaultClient` método do construtor do cliente.

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

É o mesmo que usar `standard` seguido de `build`.

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

Se você não definir explicitamente uma região usando os métodos `withRegion`, o SDK consultará a cadeia de fornecedores da região padrão para tentar determinar a região a ser usada.

Cadeia de fornecedores da região padrão

Este é o processo de pesquisa da região:

1. Qualquer região explícita definida usando-se `withRegion` ou `setRegion` no compilador propriamente dito tem precedência sobre todo o resto.
2. A variável de ambiente `AWS_REGION` está marcada. Se estiver definida, essa região será usada para configurar o cliente.

   > **Note**
   >
   > Essa variável de ambiente é definida pelo Lambda contêiner.

3. O SDK verifica o arquivo de configuração AWS compartilhado (geralmente localizado em`~/.aws/config`). Se a propriedade da região estiver presente, ela será usada pelo SDK.
   - A variável de ambiente `AWS_CONFIG_FILE` pode ser usada para personalizar o local do arquivo de configuração compartilhado.
   - A variável de ambiente `AWS_PROFILE` ou a propriedade do sistema `aws.profile` pode ser usada para personalizar o perfil carregado pelo SDK.
4. O SDK tenta usar o serviço de metadados da Amazon EC2 instância para determinar a região da instância em execução Amazon EC2 no momento.
5. Se o SDK ainda não tiver encontrado uma região a esta altura, a criação do cliente falhará com uma exceção.

Ao desenvolver AWS aplicativos, uma abordagem comum é usar o arquivo de configuração compartilhado (descrito em Usando a cadeia de fornecedores de credenciais padrão) para definir a região para o desenvolvimento local e confiar na cadeia de fornecedores da região padrão para determinar a região quando executada na AWS infraestrutura. Isso simplifica muito a criação do cliente e mantém a portabilidade do aplicativo.

Tratamento de exceções

Entender como e quando AWS SDK for Java as exceções são lançadas é importante para criar aplicativos de alta qualidade usando o SDK. As seções a seguir descrevem os casos diferentes de exceções lançadas pelo SDK e como processá-las da maneira apropriada.

Por que exceções desmarcadas?

AWS SDK for Java Ele usa exceções de tempo de execução (ou não verificadas) em vez de exceções verificadas pelos seguintes motivos:

- Como permitir que desenvolvedores controlem os erros que desejam processar sem forçá-los a processar casos excepcionais com os quais não estão preocupados (e tornar o código excessivamente detalhado)
- Para evitar problemas de escalabilidade inerentes a exceções marcadas em aplicativos grandes

Em geral, as exceções marcadas funcionam bem em escalas pequenas, mas podem se tornar problemáticas à medida que os aplicativos crescem e se tornam mais complexos.

Para obter mais informações sobre o uso de exceções marcadas e desmarcadas, consulte:

- Exceções desmarcadas – A controvérsia
- O problema com exceções marcadas
- Exceções marcadas do Java foram um equívoco (e aqui está o que eu gostaria de fazer sobre isso)

AmazonServiceException (e subclasses)

AmazonServiceExceptionné a exceção mais comum que você enfrentará ao usar AWS SDK for Java o. Essa exceção representa uma resposta de erro de um AWS service (Serviço da AWS). Por exemplo, se você tentar encerrar uma Amazon EC2 instância que não existe, EC2 retornará uma resposta de erro e todos os detalhes dessa resposta de erro serão incluídos na `AmazonServiceException` resposta lançada. Para alguns casos, uma subclasse de `AmazonServiceException` é lançada para permitir que os desenvolvedores controlem casos de erro por meio de blocos catch.

Ao encontrar um`AmazonServiceException`, você sabe que sua solicitação foi enviada com sucesso para o AWS service (Serviço da AWS) , mas não pôde ser processada com sucesso. Isso pode ocorrer devido a erros nos parâmetros da solicitação ou problemas no lado do serviço.

`AmazonServiceException` fornece informações como:

- Código de status HTTP retornado
- Código de AWS erro retornado
- Mensagem de erro detalhada do serviço
- AWS ID de solicitação para a solicitação que falhou

`AmazonServiceException`também inclui informações sobre se a solicitação com falha foi culpa do chamador (uma solicitação com valores ilegais) ou culpa dele (um erro interno AWS service (Serviço da AWS) do serviço).

AmazonClientException

AmazonClientExceptionindica que ocorreu um problema dentro do código do cliente Java, ao tentar enviar uma solicitação para AWS ou ao tentar analisar uma resposta de AWS. Um geralmente `AmazonClientException` é mais grave do que um `AmazonServiceException` e indica um grande problema que está impedindo o cliente de fazer chamadas de serviço para AWS os serviços. Por exemplo, AWS SDK for Java ele lança uma, `AmazonClientException` se nenhuma conexão de rede estiver disponível, quando você tenta chamar uma operação em um dos clientes.

Programação assíncrona

Você pode usar métodos síncronos ou assíncronos para chamar operações em serviços. AWS Os métodos síncronos bloqueiam a execução do seu thread até o cliente receber uma resposta do serviço. Os métodos assíncronos retornam imediatamente, devolvendo o controle ao thread de chamada sem aguardar uma resposta.

Como um método assíncrono retorna antes de uma resposta estar disponível, você precisa de uma maneira de obter a resposta quando ela estiver pronta. O AWS SDK for Java fornece duas formas: objetos futuros e métodos de retorno de chamada.

Futures do Java

Os métodos assíncronos AWS SDK for Java retornam um objeto Future que contém os resultados da operação assíncrona no futuro.

Chame o método `Future isDone()` para ver se o serviço já forneceu um objeto de resposta. Quando a resposta estiver pronta, você poderá obter o objeto de resposta chamando o método `Future get()`. É possível usar esse mecanismo para sondar periodicamente os resultados da operação assíncrona, enquanto o aplicativo continua funcionando em outras atividades.

Aqui está um exemplo de uma operação assíncrona que chama uma Lambda função, recebendo uma `Future` que pode conter um objeto. InvokeResult O objeto `InvokeResult` será recuperado somente depois que `isDone()` for `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);
    }
}
```

Retornos de chamada assíncronos

Além de usar o `Future` objeto Java para monitorar o status das solicitações assíncronas, o SDK também permite que você implemente uma classe que usa a interface. AsyncHandler `AsyncHandler`fornece dois métodos que são chamados dependendo de como a solicitação foi concluída: `onSuccess` `onError` e.

A principal vantagem da interface da abordagem de interface do retorno de chamada é que ela evita a necessidade de sondar o objeto `Future` para descobrir quando a requisição foi concluída. Em vez disso, o código pode iniciar imediatamente a próxima atividade e contar com o SDK para chamar o handler no momento certo.

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

Práticas recomendadas

Execução do retorno de chamada

A implementação de `AsyncHandler` é executada dentro do grupo de threads de propriedade do cliente assíncrono. Resumidamente, o código executado rapidamente é mais apropriado dentro da implementação `AsyncHandler`. Executar por muito tempo ou bloquear código dentro dos métodos handler pode causar contenção do grupo de threads usado pelo cliente assíncrono e evitar que o cliente execute requisições. Se você tiver uma tarefa de longa duração que precise começar de um retorno de chamada, o retorno de chamada deverá executar a tarefa em um novo thread ou em um grupo de threads gerenciado pelo aplicativo.

Configuração do grupo de threads

Os clientes assíncronos no AWS SDK for Java fornecem um pool de threads padrão que deve funcionar para a maioria dos aplicativos. Você pode implementar um personalizado ExecutorServicee passá-lo para clientes AWS SDK for Java assíncronos para ter mais controle sobre como os grupos de threads são gerenciados.

Por exemplo, você pode fornecer uma `ExecutorService` implementação que usa um personalizado ThreadFactorypara controlar como os encadeamentos no pool são nomeados ou para registrar informações adicionais sobre o uso do encadeamento.

Processo assíncrono

A TransferManagerclasse no SDK oferece suporte assíncrono para trabalhar com. Amazon S3`TransferManager`gerencia uploads e downloads assíncronos, fornece relatórios detalhados sobre o progresso das transferências e oferece suporte a retornos de chamada em diferentes eventos.

Registrando AWS SDK for Java chamadas

O AWS SDK for Java é instrumentado com o Apache Commons Logging, que é uma camada de abstração que permite o uso de qualquer um dos vários sistemas de registro em tempo de execução.

Entre os sistemas de registro em log compatíveis estão o Java Logging Framework e o Apache Log4j, entre outros. Este tópico mostra como usar o Log4j. Você pode usar a funcionalidade de registro em log do SDK sem fazer alterações no código do aplicativo.

Para saber mais sobre o Log4j, consulte o site do Apache.

> **Note**
>
> Este tópico se concentra no Log4j 1.x. O Log4j2 não oferece suporte diretamente ao Apache Commons Logging, mas fornece um adaptador que direciona automaticamente chamadas de registro em log para o Log4j2 usando a interface do Apache Commons Logging. Para obter mais informações, consulte Commons Logging Bridge na documentação do Log4j2.

Fazer download do Log4J JAR

Para usar o Log4j com o SDK, você precisa fazer download do Log4j JAR no site do Apache. O SDK não inclui o JAR. Copie o arquivo JAR para um local no classpath.

O Log4j usa um arquivo de configuração, log4j.properties. Os arquivos de configuração de exemplo são mostrados abaixo. Copie esse arquivo de configuração para um diretório no classpath. O Log4j JAR e o arquivo log4j.properties não precisam estar no mesmo diretório.

O arquivo de configuração log4j.properties especifica propriedades como nível de registro em log, em que a saída do registro em log é enviada (por exemplo, para um arquivo ou para o console) e o formato da saída. O nível de registro em log é a granularidade de saída gerada pelo registrador em log. O Log4j dá suporte ao conceito de várias hierarquias de registro em log. O nível de registro em log é definido de maneira independente para cada hierarquia. As duas seguintes hierarquias de registro em log estão disponíveis no AWS SDK for Java:

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

Definir o classpath

O Log4j JAR e o arquivo log4j.properties devem estar no classpath. Se você estiver usando o Apache Ant, defina o classpath no elemento `path` do arquivo Ant. O exemplo a seguir mostra um elemento de caminho do arquivo Ant para o exemplo do Amazon S3 incluído no SDK.

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

Se estiver usando o IDE do Eclipse, você poderá definir o classpath abrindo o menu e navegando até Project (Projeto) | Properties (Propriedades) | Java Build Path.

Erros e avisos específicos do serviço

Recomendamos sempre deixar a hierarquia do registrador em log "com.amazonaws" definida como "WARN" para interceptar todas as mensagens importantes das bibliotecas de cliente. Por exemplo, se o Amazon S3 cliente detectar que seu aplicativo não fechou corretamente `InputStream` e pode estar vazando recursos, o cliente S3 reportará isso por meio de uma mensagem de aviso aos registros. Isso também garante que as mensagens serão registradas em log se o cliente enfrentar algum problema ao processar requisições ou respostas.

O arquivo log4j.properties a seguir define o `rootLogger` como WARN, o que faz mensagens de erro e aviso de todos os registradores em log na hierarquia "com.amazonaws" serem incluídas. Você também pode definir explicitamente o registrador em log com.amazonaws como 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
```

Registro em log do resumo de requisição/resposta

Cada solicitação para um AWS service (Serviço da AWS) gera um ID de AWS solicitação exclusivo que é útil se você tiver problemas com a forma como um AWS service (Serviço da AWS) está lidando com uma solicitação. AWS IDs as solicitações podem ser acessadas programaticamente por meio de objetos de exceção no SDK para qualquer chamada de serviço com falha e também podem ser relatadas por meio do nível de registro DEBUG no registrador “com.amazonaws.request”.

O arquivo log4j.properties a seguir permite um resumo das solicitações e respostas, incluindo a solicitação. AWS IDs

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

Aqui está um exemplo da saída do 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
```

Registro em log detalhado

Em alguns casos, pode ser útil ver as solicitações e respostas exatas que eles AWS SDK for Java enviam e recebem. Você não deve habilitar esse registro em sistemas de produção porque escrever grandes solicitações (por exemplo, um arquivo sendo carregado Amazon S3) ou respostas pode reduzir significativamente a velocidade de um aplicativo. Se você realmente precisar acessar essas informações, poderá ativá-las temporariamente por meio do registrador Apache HttpClient 4. Habilitar o nível DEBUG no registrador em log `org.apache.http.wire` permite o registro em log de todos os dados de requisição e resposta.

O arquivo log4j.properties a seguir ativa o registro completo de conexões no Apache HttpClient 4 e só deve ser ativado temporariamente, pois pode ter um impacto significativo no desempenho do seu aplicativo.

```
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
```

Registro de métricas de latência

Se você estiver solucionando problemas e desejar ver métricas, como qual processo está consumindo mais tempo ou se o lado do servidor ou do cliente têm mais latência, o registrador de latência será útil. Defina o registrador `com.amazonaws.latency` como DEBUG para habilitá-lo.

> **Note**
>
> O registrador estará disponível somente se as métricas do SDK estiverem habilitadas. Para saber mais sobre o pacote de métricas do SDK, consulte Habilitar métricas para o 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
```

Aqui está um exemplo da saída do 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]
```

Configuração do cliente

O AWS SDK for Java permite que você altere a configuração padrão do cliente, o que é útil quando você deseja:

- Conectar-se à Internet por meio de proxy
- Alterar configurações de transporte HTTP, como tempo limite da conexão e novas tentativas de requisição
- Especificar dicas de tamanho do buffer de soquete TCP

Configuração do proxy

Ao construir um objeto cliente, você pode passar um ClientConfigurationobjeto opcional para personalizar a configuração do cliente.

Se você se conectar à Internet por meio de um servidor de proxy, será necessário configurar as definições do servidor de proxy (host do proxy, porta e nome de usuário/senha) por meio do objeto `ClientConfiguration`.

Configuração do transporte HTTP

Você pode configurar várias opções de transporte HTTP usando o ClientConfigurationobjeto. Ocasionalmente, novas opções são adicionadas; para ver a lista completa de opções que você pode recuperar ou definir, consulte a Referência da AWS SDK for Java API.

> **Note**
>
> Cada um dos valores configuráveis tem um valor padrão definido por uma constante. Para obter uma lista dos valores constantes para`ClientConfiguration`, consulte Valores de campo constantes na Referência da AWS SDK for Java API.

Conexões máximas

Você pode definir o número máximo permitido de conexões HTTP abertas usando ClientConfigurationo. setMaxConnectionsmétodo.

> **Important**
>
> Defina o número máximo de conexões para o número de transações simultâneas de modo a evitar disputas de conexão e baixo desempenho. Para o valor máximo padrão das conexões, consulte Valores de campo constantes na Referência AWS SDK for Java da API.

Tempos limite e processamento de erros

Você pode definir opções relacionadas a tempos limite e processamento de erros com conexões HTTP.

- Tempo limite da conexão

  Tempo limite da conexão é o tempo (em milissegundos) que a conexão HTTP aguardará para estabelecer uma conexão antes de desistir. O padrão é 10.000 ms.

  Para definir esse valor você mesmo, use ClientConfigurationo. setConnectionTimeoutmétodo.

- Time to Live (TTL – Tempo de vida) da conexão

  Por padrão, o SDK tentará reutilizar conexões HTTP, enquanto isso for possível. Em situações de falha nas quais uma conexão seja estabelecida com um servidor fora de serviço, ter um TTL finito pode ajudar na recuperação do aplicativo. Por exemplo, definir um TTL de 15 minutos garantirá que, mesmo se tiver uma conexão estabelecida com um servidor que esteja enfrentando problemas, você restabelecerá uma conexão com um novo servidor dentro de 15 minutos.

  Para definir o TTL da conexão HTTP, use o ClientConfigurationmétodo.setConnectionTTL.

- Máximo de repetições com erro

  A contagem máxima padrão de novas tentativas para erros repetíveis é três. Você pode definir um valor diferente usando ClientConfigurationo. setMaxErrorMétodo de repetição.

Endereço local

Para definir o endereço local ao qual o cliente HTTP se vinculará, useClientConfiguration.setLocalAddress.

Dicas de tamanho do buffer de soquete TCP

Usuários avançados que desejam ajustar parâmetros TCP de baixo nível também podem definir dicas de tamanho do buffer TCP por meio do objeto. ClientConfiguration A maioria dos usuários jamais precisará ajustar esses valores, mas eles são fornecidos para usuários avançados.

Os tamanhos de buffer TCP ideais de um aplicativo dependem muito das configurações e dos recursos da rede e do sistema operacional. Por exemplo, a maioria dos sistemas operacionais modernos oferece uma lógica de autoajuste para tamanhos de buffer TCP. Isso pode ter um grande impacto sobre o desempenho para conexões TCP mantidas abertas pelo tempo necessário para o autoajuste otimizar tamanhos de buffer.

Tamanhos de buffer grandes (por exemplo, 2 MB) permitem que o sistema operacional armazene em buffer mais dados na memória sem exigir que o servidor remoto confirme o recebimento dessas informações e, assim, podem ser especialmente úteis quando a rede tem alta latência.

Trata-se apenas de uma dica, e o sistema operacional talvez não esteja apto para isso. Ao usar essa opção, os usuários devem sempre verificar os limites configurados do sistema operacional e os padrões. A maioria dos sistemas operacionais tem um limite de tamanho de buffer TCP máximo configurado e não permitirá ir além desse limite, a menos que você aumente explicitamente o limite do tamanho de buffer TCP máximo.

Muitos recursos estão disponíveis para ajudar a definir configurações de tamanhos do buffer TCP e configurações específicas do sistema operacional, inclusive o seguinte:

- Ajuste do host

Políticas de controle de acesso

AWS as políticas de controle de acesso permitem que você especifique controles de acesso refinados em seus recursos. AWS Uma política de controle de acesso consiste em um conjunto de instruções, que assumem a forma:

A conta A tem permissão para realizar a ação B no recurso C em que a condição D se aplica.

Em que:

- A é o principal - Conta da AWS Aquele que está fazendo uma solicitação para acessar ou modificar um de seus AWS recursos.
- B é a ação: a maneira pela qual seu AWS recurso está sendo acessado ou modificado, como enviar uma mensagem para uma Amazon SQS fila ou armazenar um objeto em um Amazon S3 bucket.
- C é o recurso - A AWS entidade que o principal deseja acessar, como uma Amazon SQS fila ou um objeto armazenado Amazon S3.
- D é um conjunto de condições: as restrições opcionais que especificam quando permitir ou negar acesso para a entidade principal acessar o recurso. Muitas condições expressivas estão disponíveis, algumas específicas de cada serviço. Por exemplo, você pode usar condições de data para permitir acesso aos recursos somente depois ou antes de uma hora específica.

Amazon S3 Exemplo

O exemplo a seguir demonstra uma política que permite que qualquer pessoa acesse para ler todos os objetos em um bucket, mas restringe o acesso ao upload de objetos nesse bucket a dois Conta da AWS s específicos (além da conta do proprietário do 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 Exemplo

Um uso comum das políticas é autorizar uma Amazon SQS fila para receber mensagens de um tópico do 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));
```

Exemplo do Amazon SNS

Alguns serviços oferecem condições adicionais que podem ser usadas em políticas. O Amazon SNS fornece condições para permitir ou negar assinaturas de tópicos do SNS com base no protocolo (por exemplo, e-mail, HTTP, HTTPS Amazon SQS) e no endpoint (por exemplo, endereço de e-mail, URL, Amazon SQS ARN) da solicitação de assinatura de um tópico.

```
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()));
```

Definir o JVM TTL para pesquisas de nome DNS

A JVM armazena em cache pesquisas de nome DNS. Quando a JVM resolve um nome de host para um endereço IP, ela armazena o endereço IP em cache por um período de tempo especificado, conhecido como (TTL). time-to-live

Como AWS os recursos usam entradas de nome DNS que mudam ocasionalmente, recomendamos que você configure sua JVM com um valor TTL de 5 segundos. Isso garante que, quando o

endereço IP de um recurso mudar, o aplicativo poderá receber e usar o novo endereço IP do recurso consultando novamente o DNS.

Em algumas configurações do Java, o TTL padrão da JVM é definido de maneira que jamais atualizará entradas DNS até a JVM ser reiniciada. Portanto, se o endereço IP de um AWS recurso mudar enquanto seu aplicativo ainda estiver em execução, ele não poderá usar esse recurso até que você reinicie manualmente a JVM e as informações IP em cache sejam atualizadas. Nesse caso, é crucial definir o TTL da JVM, de forma que ele atualize periodicamente as informações de IP armazenadas em cache.

Como definir o TTL da JVM

Para modificar o TTL da JVM, defina o valor da propriedade de segurança networkaddress.cache.ttl, defina a propriedade `networkaddress.cache.ttl` no arquivo `$JAVA_HOME/jre/lib/security/java.security` para Java 8 ou arquivo `$JAVA_HOME/conf/security/java.security` para Java 11 ou posterior.

Veja a seguir um trecho de um arquivo `java.security` que mostra o cache de TTL definido para 5 segundos.

```
#
# 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
...
```

Todas as aplicações executadas na JVM representada pela variável de ambiente `$JAVA_HOME` usam essa configuração.

Habilitando métricas para o AWS SDK for Java

Eles AWS SDK for Java podem gerar métricas para visualização e monitoramento com a Amazon CloudWatch que medem:

- o desempenho do seu aplicativo ao acessar AWS
- o desempenho do seu JVMs quando usado com AWS
- detalhes do ambiente do tempo de execução, como a memória do heap, o número de threads e os descritores de arquivo aberto

Como habilitar a geração de métricas do SDK do Java

Você precisa adicionar a seguinte dependência do Maven para permitir que o SDK envie métricas para. 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>
```

*Substitua o número da versão pela versão mais recente do SDK disponível na central do Maven.

AWS SDK for Java as métricas são desativadas por padrão. Para habilitá-la para o ambiente de desenvolvimento local, inclua uma propriedade de sistema que aponte para o arquivo de credencial de segurança da AWS durante a inicialização da JVM. Por exemplo:

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

Você precisa especificar o caminho para seu arquivo de credencial para que o SDK possa carregar os pontos de dados coletados para CloudWatch análise posterior.

> **Note**
>
> Se você estiver acessando AWS de uma Amazon EC2 instância usando o serviço de metadados da Amazon EC2 instância, não precisará especificar um arquivo de credencial. Neste caso, você precisa especificar somente:
>
> ```
> -Dcom.amazonaws.sdk.enableDefaultMetrics
> ```

Todas as métricas capturadas pelo AWS SDK for Java estão no namespace AWSSDK/Java e são enviadas para a região CloudWatch padrão (us-east-1). Para alterar a região, especifique-a usando o atributo `cloudwatchRegion` na propriedade do sistema. Por exemplo, para definir a CloudWatch região como us-east-1, use:

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

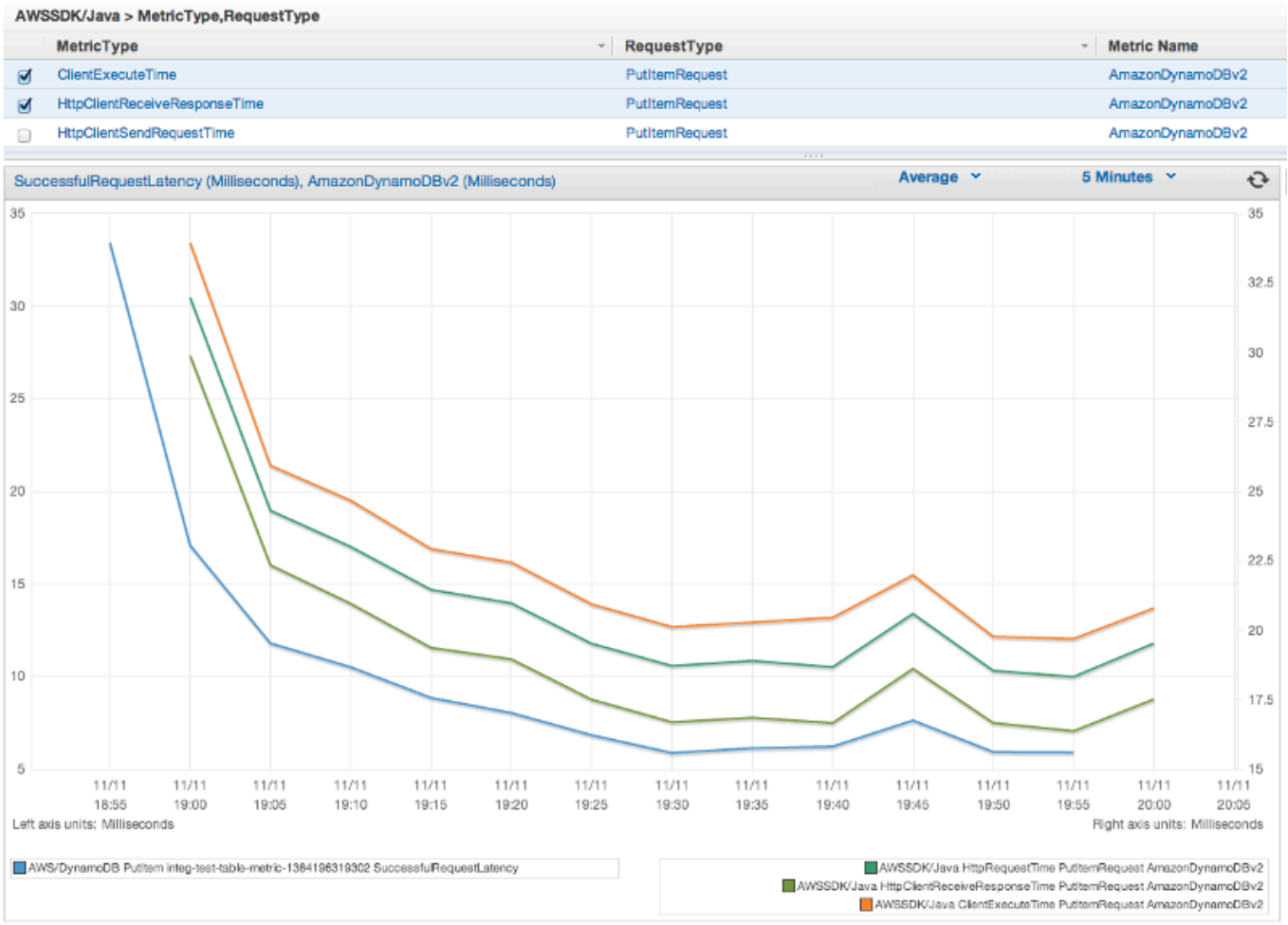
Depois de ativar o recurso, toda vez que houver uma solicitação de serviço para o AWS SDK for Java, pontos AWS de dados métricos serão gerados, colocados em fila para um resumo estatístico e enviados de forma assíncrona para aproximadamente uma vez a CloudWatch cada minuto. Assim que o upload das métricas for feito, você poderá visualizá-las usando o Console de gerenciamento da AWS e definir alarmes para problemas em potencial, como vazamento de memória, vazamento do descritor de arquivo etc.

Tipos de métrica disponíveis

O conjunto padrão de métricas é dividido em três categorias principais:

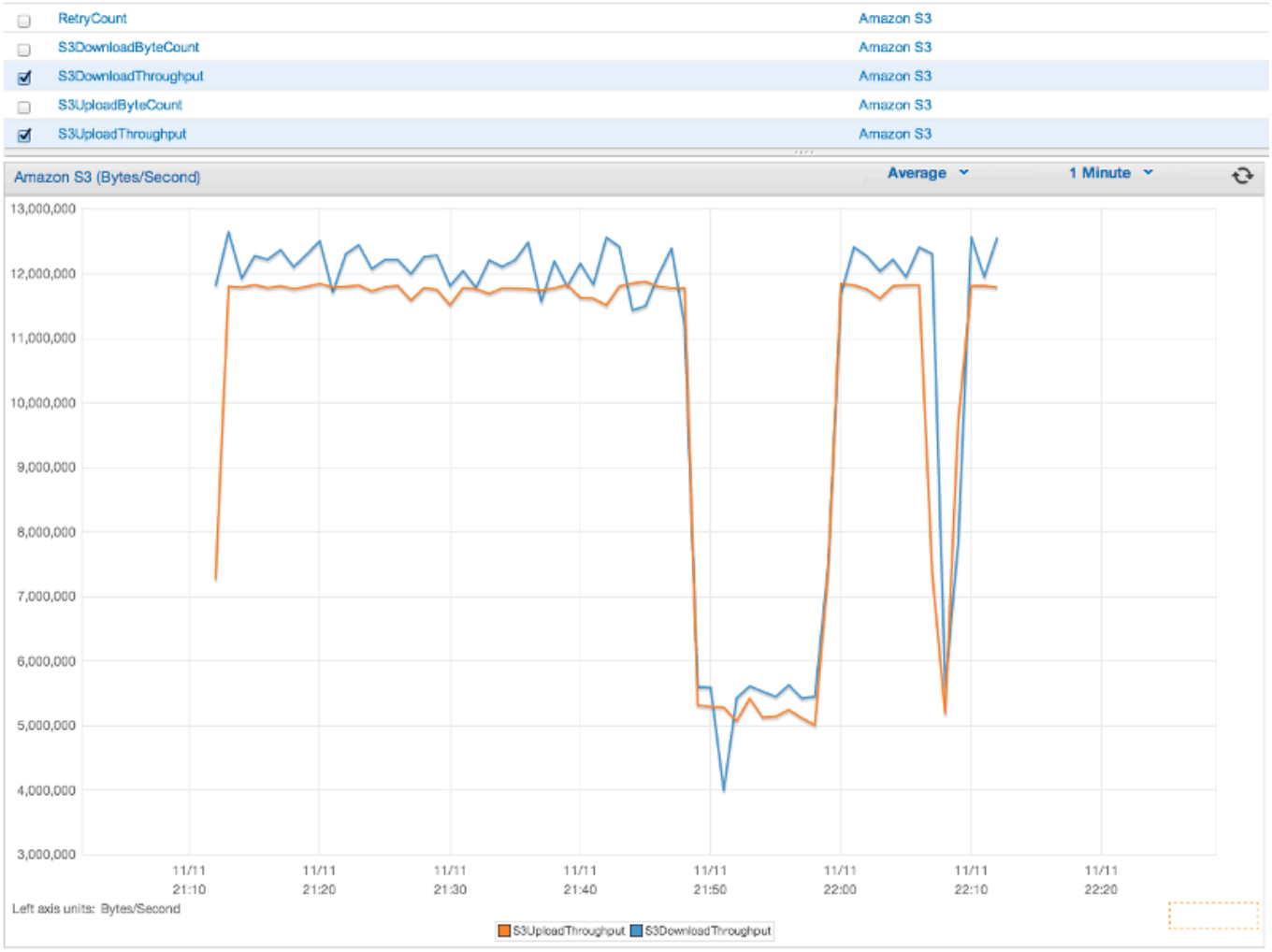
AWS Métricas de solicitação

- Abrange áreas como a latência da requisição/resposta HTTP, o número de requisições, exceções e novas tentativas.



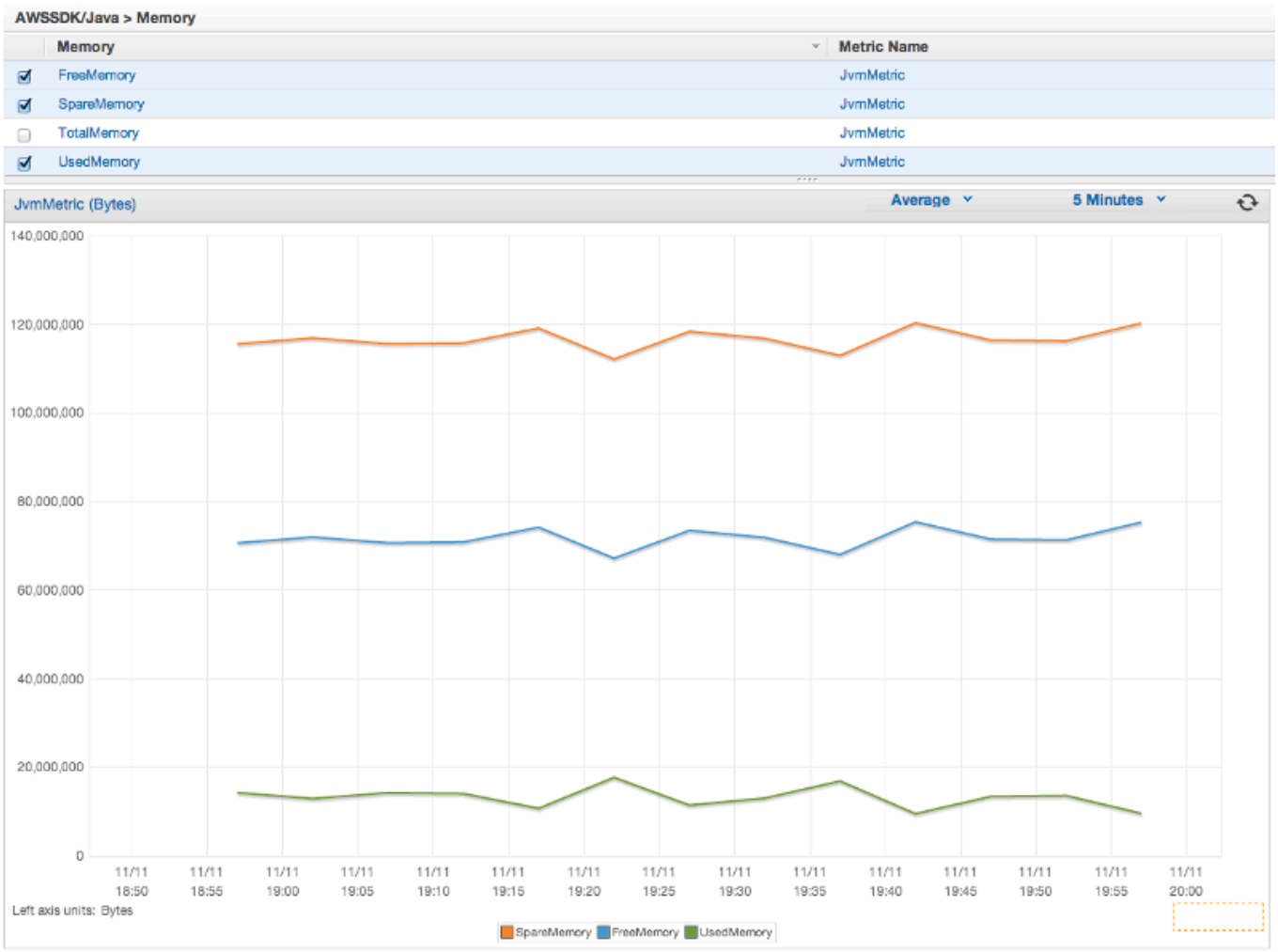
AWS service (Serviço da AWS) Métricas

- Inclua dados AWS service (Serviço da AWS) específicos, como a taxa de transferência e a contagem de bytes para uploads e downloads do S3.



Métricas de máquina

- Abrangem o ambiente do tempo de execução, inclusive a memória do heap, o número de threads e os descritores de arquivo aberto.



Se você quiser excluir métricas de máquina, adicione `excludeMachineMetrics` à propriedade do sistema:

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

Mais informações

- Consulte amazonaws/metrics package summary para obter uma lista completa dos tipos de métrica de núcleo predefinidos.
- Saiba mais sobre como trabalhar com o CloudWatch uso do AWS SDK for Java em CloudWatch Exemplos usando AWS SDK for Java o.

- Saiba mais sobre o ajuste de desempenho na postagem do blog Tuning AWS SDK for Java to Improve Resiliency.

AWS SDK for Java Code Examples

Esta seção fornece tutoriais e exemplos de como usar o AWS SDK for Java v1 para programar os serviços da AWS.

Encontre o código-fonte para esses exemplos e outros no repositório de exemplos de código no GitHub da documentação da AWS.

Para sugerir um novo exemplo de código para a equipe de documentação da AWS considerar a produção, crie uma solicitação. A equipe está buscando produzir exemplos de código que abrangem cenários e casos de uso mais amplos, em vez de trechos de código simples que abrangem apenas chamadas de API individuais. Para obter instruções, consulte as Diretrizes de contribuição no repositório de exemplos de código no GitHub.

AWS SDK for Java 2.x

Em 2018, a AWS lançou o AWS SDK for Java 2.x. Este guia contém instruções sobre como usar o SDK para Java mais recente junto com um código de exemplo.

> **Note**
>
> Consulte Documentação e recursos adicionais para obter mais exemplos e recursos adicionais disponíveis para desenvolvedores do AWS SDK for Java!

Exemplos do CloudWatch usando o AWS SDK for Java

Esta seção apresenta exemplos de como programar o CloudWatch usando o AWS SDK for Java.

O Amazon CloudWatch monitora os recursos da Amazon Web Services (AWS) e as aplicações que você executa na AWS em tempo real. Você pode usar o CloudWatch para coletar e monitorar métricas, que são as variáveis mensuráveis que ajudam você a avaliar seus recursos e aplicativos. Os alarmes do CloudWatch enviam notificações ou fazem alterações automaticamente nos recursos que você está monitorando com base nas regras definidas.

Para obter mais informações sobre o CloudWatch, consulte o Guia do usuário do Amazon CloudWatch.

> **Note**
>
> Entre os exemplos está somente o código necessário para demonstrar cada técnica. O código de exemplo completo está disponível no GitHub. A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

Obter métricas do CloudWatch

Listar métricas

Para listar métricas do CloudWatch, crie um ListMetricsRequest e chame o método `listMetrics` do AmazonCloudWatchClient. Você pode usar o `ListMetricsRequest` para filtrar as métricas retornadas por namespace, nome da métrica ou dimensões.

> **Note**
>
> Uma lista de métricas e dimensões publicadas pelos serviços da AWS pode ser encontrada em {https---docs-aws-amazon-com-AmazonCloudWatch-Latest-Monitoring-CW-Support-for-AWS-html} [Referência de métricas e dimensões do Amazon CloudWatch] no Guia do usuário do Amazon CloudWatch.

Importações

```
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;
```

Código da

```
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;
    }
}
```

As métricas são retornadas em um ListMetricsResult chamando o método `getMetrics`.
Os resultados podem ser paginados. Para recuperar o próximo lote de resultados, chame `setNextToken` no objeto de solicitação original com o valor de retorno do método `ListMetricsResult` do objeto `getNextToken` e passe o objeto de solicitação modificado para outra chamada para `listMetrics`.

Mais informações

- ListMetrics na Referência de API do Amazon CloudWatch.

Publicar dados de métrica personalizada

Vários serviços da AWS publicam as próprias métricas em namespaces que começam com "AWS". Também é possível publicar dados de métricas personalizadas usando seu próprio namespace (contanto que não comece com "AWS").

Publicar dados de métrica personalizada

Para publicar os próprios dados de métrica, chame o método `putMetricData` do AmazonCloudWatchClient com um PutMetricDataRequest. O `PutMetricDataRequest` deve incluir o namespace personalizado a ser usado para os dados e as informações sobre o próprio ponto de dados em um objeto MetricDatum.

> **Note**
>
> Você não pode especificar um namespace que começa com "AWS". Namespaces que começam com "AWS" são reservados para serem usados por produtos da Amazon Web Services.

Importações

```
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;
```

Código da

```
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);
```

Mais informações

- Usar métricas do Amazon CloudWatch no Guia do usuário do Amazon CloudWatch.
- Namespaces da AWS no Guia do usuário Amazon CloudWatch.
- PutMetricData na Referência de API do Amazon CloudWatch.

Trabalhar com alarmes do CloudWatch

Criar um alarme

Para criar um alarme com base em uma métrica do CloudWatch, chame o método `putMetricAlarm` do AmazonCloudWatchClient com um PutMetricAlarmRequest preenchido com as condições de alarme.

Importações

```
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;
```

Código da

```
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);
```

Listar alarmes

Para listar os alarmes do CloudWatch criados por você, chame o método `describeAlarms` do AmazonCloudWatchClient com um DescribeAlarmsRequest que pode ser usado para definir opções para o resultado.

Importações

```
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;
```

Código da

```
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;
    }
}
```

A lista de alarmes pode ser obtida chamando `getMetricAlarms` no DescribeAlarmsResult retornado por `describeAlarms`.

Os resultados podem ser paginados. Para recuperar o próximo lote de resultados, chame `setNextToken` no objeto de solicitação original com o valor de retorno do método `DescribeAlarmsResult` do objeto `getNextToken` e passe o objeto de solicitação modificado para outra chamada para `describeAlarms`.

> **Note**
>
> Você também pode recuperar alarmes para uma métrica específica usando o método `describeAlarmsForMetric` do AmazonCloudWatchClient. O uso é semelhante a `describeAlarms`.

Excluir alarmes

Para excluir os alarmes do CloudWatch, chame o método `deleteAlarms` do AmazonCloudWatchClient com um DeleteAlarmsRequest contendo um ou mais nomes de alarmes que você deseja excluir.

Importações

```
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;
```

Código da

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

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

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

Mais informações

- Criar alarmes do Amazon CloudWatch, no Guia do usuário do Amazon CloudWatch
- PutMetricAlarm na Referência de API do Amazon CloudWatch
- DescribeAlarms na Referência de API do Amazon CloudWatch
- DeleteAlarms na Referência de API do Amazon CloudWatch

Usar ações de alarme no CloudWatch

Usando ações de alarme do CloudWatch, é possível criar alarmes que realizam ações como interromper, encerrar, reinicializar ou recuperar automaticamente instâncias do Amazon EC2.

> **Note**
>
> As ações de alarme podem ser adicionadas a um alarme usando-se o método de PutMetricAlarmRequest`setAlarmActions` quando se cria um alarme.

Habilitar ações de alarme

Para habilitar ações de um alarme do CloudWatch, chame o `enableAlarmActions` do AmazonCloudWatchClient com um EnableAlarmActionsRequest que contém um ou mais nomes de alarmes cujas ações você deseja habilitar.

Importações

```
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;
```

Código da

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

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

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

Desabilitar ações de alarme

Para desabilitar ações de um alarme do CloudWatch, chame o `disableAlarmActions` do AmazonCloudWatchClient com um DisableAlarmActionsRequest que contém um ou mais nomes de alarmes cujas ações você deseja desabilitar.

Importações

```
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;
```

Código da

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

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

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

Mais informações

- Criar alarmes para interromper, encerrar, reinicializar ou recuperar uma instância no Guia do Usuário do Amazon CloudWatch

- PutMetricAlarm na Referência de API do Amazon CloudWatch
- EnableAlarmActions na Referência de API do Amazon CloudWatch
- DisableAlarmActions na Referência de API do Amazon CloudWatch

Enviar eventos do ao CloudWatch

O CloudWatch Events distribui um fluxo quase em tempo real de eventos do sistema que descrevem alterações feitas em recursos da AWS para instâncias do Amazon EC2, funções do Lambda, fluxos do Kinesis, tarefas do Amazon ECS, máquinas de estado do Step Functions, tópicos do Amazon SNS, filas do Amazon SQS ou destinos internos. Você pode comparar eventos e roteá-los para um ou mais fluxos ou funções de destino usando regras simples.

Adicionar eventos

Para adicionar eventos do CloudWatch personalizados, chame o método `putEvents` do AmazonCloudWatchEventsClient com um objeto PutEventsRequest que contenha um ou mais objetos PutEventsRequestEntry que fornecem detalhes sobre cada evento. Você pode especificar vários parâmetros para a entrada, como a origem e o tipo do evento, recursos associados ao evento e assim por diante.

> **Note**
>
> Você pode especificar um máximo de dez eventos por chamada para `putEvents`.

Importações

```
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;
```

Código da

```
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);
```

Adicionar regras

Para criar ou atualizar uma regra, chame o método `putRule` do AmazonCloudWatchEventsClient com um PutRuleRequest com o nome da regra e os parâmetros opcionais, como o padrão de evento, o perfil do IAM a ser associado à regra e uma expressão de programação que descreva com que frequência a regra é executada.

Importações

```
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;
```

Código da

```
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);
```

Adicionar destinos

Destinos são os recursos invocados quando uma regra é disparada. Entre os destinos de exemplo estão instâncias do Amazon EC2, funções do Lambda, streamings do Kinesis, tarefas do Amazon ECS, máquinas de estado do Step Functions e destinos integrados.

Para adicionar um destino a uma regra, chame o método `putTargets` do AmazonCloudWatchEventsClient com um PutTargetsRequest que contenha a regra a ser atualizada e uma lista de destinos a serem adicionados à regra.

Importações

```
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;
```

Código da

```
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);
```

Mais informações

- Adicionar eventos com PutEvents no Guia do Usuário do Amazon CloudWatch Events
- Programar expressões para regras no Guia do Usuário do Amazon CloudWatch Events
- Tipos de eventos para o CloudWatch Events no Guia do usuário do Amazon CloudWatch Events
- Eventos e padrões de evento no Guia do Usuário do Amazon CloudWatch Events

- PutEvents na Referência de API do Amazon CloudWatch Events
- PutTargets na Referência de API do Amazon CloudWatch Events
- PutRule na Referência de API do Amazon CloudWatch Events

DynamoDB Exemplos de usando a AWS SDK for Java

Esta seção apresenta exemplos de como programar o DynamoDB usando o AWS SDK for Java.

> **Note**
>
> Entre os exemplos está somente o código necessário para demonstrar cada técnica. O código de exemplo completo está disponível no GitHub. A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

Usar endpoints baseados em conta da AWS

O DynamoDB oferece endpoints baseados em contas da AWS que podem melhorar o desempenho usando seu ID de conta da AWS para simplificar o roteamento de solicitações.

Para aproveitar esse recurso, use a versão 1.12.771 ou posterior da versão 1 do AWS SDK for Java. É possível encontrar a versão mais recente do SDK listado norepositório central do Maven. Depois que uma versão compatível do SDK está ativa, os novos endpoints são usados automaticamente.

Se quiser optar por não utilizar o roteamento baseado em contas, você terá quatro opções:

- Configurar um cliente de serviço do DynamoDB com o `AccountIdEndpointMode` definido como `DISABLED`.
- Definir uma variável de ambiente.
- Definir uma propriedade do sistema da JVM.

- Atualizar a definição do arquivo de configuração compartilhado AWS.

O seguinte trecho é um exemplo de como desabilitar o roteamento baseado em contas configurando um cliente de serviço do DynamoDB:

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

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

O Guia de referência e ferramentas de AWS SDKs fornece mais informações sobre as últimas três opções de configuração.

Trabalho com tabelas no DynamoDB

Tabelas são os contêineres de todos os itens em um banco de dados do DynamoDB. Para adicionar ou remover dados do DynamoDB, você deve criar uma tabela.

Para cada tabela, você deve definir:

- Um nome de tabela é exclusivo para a conta e a região.
- Uma chave primária para a qual cada valor deve ser único; dois itens na tabela não podem ter o mesmo valor de chave primária.

  Uma chave primária pode ser simples, consistindo em uma única chave de partição (HASH) ou composta, que consiste em uma partição e uma chave de classificação (RANGE).

  Cada valor de chave tem um tipo de dados associado, enumerados pela classe ScalarAttributeType. O valor da chave pode ser binário (B), numérico (N) ou uma string (S). Para obter mais informações, consulte Regras de nomenclatura e tipos de dados no Guia do desenvolvedor do Amazon DynamoDB.
- Valores de throughput provisionado que definem o número de unidades de capacidade de leitura/gravação reservadas para a tabela.

> **Note**
>
> A definição de preço do Amazon DynamoDB se baseia nos valores de throughput provisionados definidos por você nas tabelas. Dessa forma, reserve somente a capacidade máxima de que você imagina precisar para a tabela.

O throughput provisionado para uma tabela pode ser modificado a qualquer momento. Dessa forma, você poderá ajustar a capacidade se as necessidades mudarem.

Criar uma tabela

Use o método `createTable` do cliente do DynamoDB para criar uma nova tabela do DynamoDB. Você precisa construir atributos de tabela e um esquema de tabela, ambos usados para identificar a chave primária da tabela. Você também deve fornecer valores de throughput provisionado iniciais e um nome de tabela. Defina atributos de tabela de chaves apenas ao criar sua tabela de DynamoDB.

> **Note**
>
> Se uma tabela com o nome escolhido por você já existir, um AmazonServiceException será lançado.

Importações

```
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;
```

Criar uma tabela com uma chave primária simples

Este código cria uma tabela com uma chave primária simples ("Name").

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Criar uma tabela com uma chave primária composta

Adicione outro AttributeDefinition e KeySchemaElement a CreateTableRequest.

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Listar tabelas

Você pode listar as tabelas em uma determinada região chamando o método `listTables` do cliente do DynamoDB.

> **Note**
>
> Se a tabela nomeada não existir para a conta e a região, um ResourceNotFoundException será lançado.

Importações

```
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;
```

Código da

```
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;
        }
```

Por padrão, até 100 tabelas são retornadas por chamada. Use `getLastEvaluatedTableName` no objeto ListTablesResult retornado para obter a tabela mais recentemente avaliada. Você pode usar esse valor para iniciar a listagem depois do último valor retornado da listagem anterior.

Veja o exemplo completo no GitHub.

Descrever (obter informações sobre) uma tabela

Chame o método `describeTable` do cliente do DynamoDB.

> **Note**
>
> Se a tabela nomeada não existir para a conta e a região, um ResourceNotFoundException será lançado.

Importações

```
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;
```

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Modificar (atualizar) uma tabela

Você pode modificar os valores de throughput provisionado da tabela a qualquer momento chamando o método `updateTable` do cliente do DynamoDB.

> **Note**
>
> Se a tabela nomeada não existir para a conta e a região, um ResourceNotFoundException será lançado.

Importações

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

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Excluir uma tabela

Chame o método `deleteTable` do cliente do DynamoDB e passe o nome da tabela para ele.

> **Note**
>
> Se a tabela nomeada não existir para a conta e a região, um ResourceNotFoundException será lançado.

Importações

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

Código da

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

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

Veja o exemplo completo no GitHub.

Mais informações

- Diretrizes para trabalhar com tabelas no Guia do desenvolvedor do Amazon DynamoDB
- Trabalhar com tabelas no DynamoDB no Guia do Desenvolvedor do Amazon DynamoDB

Trabalho com itens no DynamoDB

No DynamoDB, um item é um conjunto de atributos, e cada um tem um nome e um valor. Um valor de atributo pode ser uma escalar, um conjunto ou um tipo de documento. Para obter mais informações, consulte Regras de nomenclatura e tipos de dados no Guia do desenvolvedor do Amazon DynamoDB.

Recuperar (obter) um item de uma tabela

Chame o método `getItem` do AmazonDynamoDB e passe um objeto GetItemRequest para ele com o nome da tabela e o valor da chave primária do item desejado. Ele retorna um objeto GetItemResult.

Você pode usar o método `getItem( )` do objeto `GetItemResult` retornado para recuperar um Mapa dos pares de chave (String) e valor (AttributeValue) associados ao item.

Importações

```
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;
```

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Adicionar um novo item a uma tabela

Crie um Mapa de pares de chave/valor que representem os atributos do item. Eles devem incluir valores para os campos de chave primária da tabela. Se o item identificado pela chave primária já existir, os campos serão atualizados pela requisição.

> **Note**
>
> Se a tabela nomeada não existir para a conta e a região, um ResourceNotFoundException será lançado.

Importações

```
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;
```

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Atualizar um item existente em uma tabela

Você pode atualizar um atributo para um item já existente em uma tabela usando o método `updateItem` do AmazonDynamoDB, fornecendo um nome de tabela, o valor da chave primária e um mapa de campos a ser atualizado.

> **Note**
>
> Se a tabela nomeada não existir para a conta e a região, ou se o item identificado pela chave primária passada não existir, uma ResourceNotFoundException será lançada.

Importações

```
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;
```

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Usar a classe DynamoDBMapper

O AWS SDK for Java fornece uma classe DynamoDBMapper, permitindo mapear classes no lado do cliente para tabelas do Amazon DynamoDB. Para usar a classe DynamoDBMapper, defina o relacionamento entre itens em uma tabela do DynamoDB e suas instâncias de objeto correspondentes no código usando anotações (conforme mostrado no exemplo de código a seguir). A classe DynamoDBMapper permite acessar tabelas, realizar várias operações de criação, leitura, atualização e exclusão (CRUD) e executar consultas.

> **Note**
>
> A classe DynamoDBMapper não permite criar, atualizar ou excluir tabelas.

Importações

```
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;
```

Código da

O exemplo de código Java a seguir demonstra como adicionar conteúdo à tabela Music (Música) usando a classe DynamoDBMapper. Depois que o conteúdo é adicionado à tabela, observe que um item é carregado usando as teclas Partition e Sort . Depois disso, o item Awards (Prêmios) é atualizado. Para obter informações sobre como criar a tabela Música, consulte Criar uma tabela no Guia do desenvolvedor do Amazon DynamoDB.

```
        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;
        }
    }
```

Veja o exemplo completo no GitHub.

Mais informações

- Diretrizes para trabalhar com itens no Guia do desenvolvedor do Amazon DynamoDB
- Trabalho com itens no DynamoDB no Guia do Desenvolvedor do Amazon DynamoDB

Amazon EC2 Exemplos de usando a AWS SDK for Java

Esta seção apresenta exemplos de como programar o Amazon EC2 com o AWS SDK for Java.

Tópicos

- Tutorial: iniciar uma instância do EC2
- Usar perfis do IAM para conceder acesso a recursos da AWS no Amazon EC2
- Tutorial: instâncias spot do Amazon EC2
- Tutorial: gerenciamento de requisições spot do Amazon EC2 avançado
- Gerenciar instâncias do Amazon EC2
- Usar endereços IP elásticos no Amazon EC2
- Usar regiões e zonas de disponibilidade
- Trabalhar com pares de chaves do Amazon EC2
- Como trabalhar com grupos de segurança no Amazon EC2

Tutorial: iniciar uma instância do EC2

Este tutorial demonstra como usar o AWS SDK for Java para iniciar uma instância do EC2.

Tópicos

Pré-requisitos

Antes de começar, verifique se você criou uma Conta da AWS e configurou as credenciais da AWS. Para obter mais informações, consulte Conceitos básicos.

Criar um security group do Amazon EC2

O EC2-Classic será descontinuado

> **Warning**
>
> Estamos aposentando o EC2-Classic em 15 de agosto de 2022. É recomendável migrar do EC2-Classic para uma VPC. Consulte mais informações na publicação de blog EC2-Classic Networking is Retiring – Here's How to Prepare.

Crie um security group, que funciona como um firewall virtual que controla o tráfego de rede para uma ou mais instâncias do EC2. Por padrão, o Amazon EC2 associa as instâncias a um security group que não permite tráfego de entrada. Você pode criar um security group que permita que as instâncias do EC2 aceitem um determinado tráfego. Por exemplo, se precisar se conectar a uma instância do Linux, você deverá configurar o security group para permitir tráfego SSH. Você pode criar um grupo de segurança usando o console do Amazon EC2 ou o AWS SDK for Java.

Um security group é criado para seu usado no EC2-Classic e EC2-VPC. Para obter mais informações sobre o EC2-Classic e o EC2-VPC, consulte Plataformas compatíveis no Manual do usuário do Amazon EC2 para instâncias do Linux.

Para obter mais informações sobre como criar um grupo de segurança usando o console do Amazon EC2, consulte Grupos de segurança do Amazon EC2 no Guia do usuário do Amazon EC2 para instâncias Linux.

1. Crie e inicialize uma instância CreateSecurityGroupRequest. Use o método withGroupName para definir o nome do grupo de segurança e o método withDescription para definir a descrição do grupo de segurança da seguinte maneira:

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

   O nome do grupo de segurança deve ser exclusivo dentro da região da AWS na qual você inicializa o cliente do Amazon EC2. Você deve usar caracteres US-ASCII para o nome e a descrição do security group.

2. Passe o objeto de requisição como um parâmetro para o método createSecurityGroup. O método retorna um objeto CreateSecurityGroupResult, conforme mostrado a seguir:

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

   Se você tentar criar um security group com o mesmo nome de um security group existente, `createSecurityGroup` lançará uma exceção.

Por padrão, um novo security group não permite tráfego de entrada para a instância do Amazon EC2. Para permitir o tráfego de entrada, você deve autorizar explicitamente a entrada no security group. Você pode autorizar a entrada para endereços IP individuais, para um intervalo de endereços IP, para um protocolo específico e para portas TCP/UDP.

1. Crie e inicialize uma instância IpPermission. Use o método withIpv4Ranges a fim definir o intervalo de endereços IP para autorizar a entrada, e use o método withIpProtocol para definir o protocolo IP. Use os métodos withFromPort e withToPort para especificar um intervalo de portas para autorizar a entrada da seguinte maneira:

   ```
   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);
```

Todas as condições especificadas por você no objeto `IpPermission` devem ser atendidas para que a entrada seja permitida.

Especifique o endereço IP usando notação CIDR. Se especificar o protocolo como TCP/UDP, você deverá fornecer uma porta de origem e uma porta de destino. Você poderá autorizar portas somente se especificar TCP ou UDP.

2. Crie e inicialize uma instância AuthorizeSecurityGroupIngressRequest. Use o método `withGroupName` para especificar o nome do grupo de segurança e transmita o objeto `IpPermission` inicializado anteriormente para o método withIpPermissions da seguinte maneira:

```
AuthorizeSecurityGroupIngressRequest authorizeSecurityGroupIngressRequest =
    new AuthorizeSecurityGroupIngressRequest();

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

3. Transmita o objeto de requisição para o método authorizeSecurityGroupIngress da seguinte maneira:

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

Se você chamar `authorizeSecurityGroupIngress` com endereços IP para os quais a entrada já esteja autorizada, o método lançará uma exceção. Crie e inicialize um novo objeto `IpPermission` para autorizar a entrada para IPs, portas e protocolos diferentes antes de chamar `AuthorizeSecurityGroupIngress`.

Sempre que você chama os métodos authorizeSecurityGroupIngress ou authorizeSecurityGroupEgress, uma regra é adicionada ao grupo de segurança.

Criar um par de chaves

Você deve especificar um par de chaves ao executar uma instância do EC2 e, em seguida, especificar a chave privada do par de chaves ao se conectar à instância. É possível criar um par de

chaves ou usar um par de chaves existente que você usou ao iniciar outras instâncias. Para obter mais informações, consulte Pares de chaves do Amazon EC2 no Guia do Usuário do Amazon EC2 para instâncias do Linux.

1. Crie e inicialize uma instância CreateKeyPairRequest. Use o método withKeyName para definir o nome do par de chaves da seguinte maneira:

```
CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();

createKeyPairRequest.withKeyName(keyName);
```

> **Important**
>
> Os nomes do par de chaves devem ser exclusivos. Se tentar criar um par de chaves com o mesmo nome de chave como um par de chaves existente, você receberá uma exceção.

2. Passe o objeto de requisição para o método createKeyPair. O método retorna uma instância CreateKeyPairResult da seguinte maneira:

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

3. Chame o método getKeyPair do objeto resultante para obter um objeto KeyPair. Chame o método getKeyMaterial do objeto `KeyPair` para obter a chave privada codificada por PEM da seguinte maneira:

```
KeyPair keyPair = new KeyPair();

keyPair = createKeyPairResult.getKeyPair();

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

Executar uma instância do Amazon EC2

Use o procedimento a seguir para executar uma ou mais instâncias do EC2 configuradas de maneira idêntica na mesma Amazon Machine Image (AMI – Imagem de máquina da Amazon). Depois de criar as instâncias do EC2, você poderá verificar o status. Depois que as instâncias do EC2 estiverem em execução, você poderá se conectar a elas.

1. Crie e inicialize uma instância de RunInstancesRequest. Verifique se a AMI, o par de chaves e o security group especificados por você existem na região especificada quando criou o objeto de cliente.

```
RunInstancesRequest runInstancesRequest =
   new RunInstancesRequest();

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

withImageId

- O ID da AMI. Para saber como encontrar AMIs públicas fornecidas pela Amazon ou criar sua própria, consulte Imagem de máquina da Amazon (AMI).

withInstanceType

- Um tipo de instância compatível com a AMI especificada. Para obter mais informações, consulte Tipos de instância no Guia do usuário do Amazon EC2 para instâncias do Linux.

withMinCount

- O número mínimo de instâncias do EC2 a serem executadas. Se isso for mais instâncias do que o Amazon EC2 pode executar na zona de disponibilidade de destino, o Amazon EC2 não executará nenhuma instância.

withMaxCount

- O número máximo de instâncias do EC2 a serem executadas. Se isso for mais instâncias do que o Amazon EC2 pode executar na zona de disponibilidade de destino, o Amazon EC2 executará o maior número possível de instâncias acima de `MinCount`. É possível executar entre 1 e o número máximo de instâncias às quais você tem permissão para o tipo de instância. Para obter mais informações, consulte Quantas instâncias posso executar no Amazon EC2 nas perguntas frequentes do Amazon EC2.

withKeyName

- O nome do par de chaves do EC2. Se você iniciar uma instância sem especificar um par de chaves, não poderá se conectar a ela. Para obter mais informações, consulte Criar um par de chaves.

withSecurityGroups

- Um ou mais security groups. Para obter mais informações, consulte Criar um grupo de segurança do Amazon EC2.

2. Execute as instâncias passando o objeto de requisição para o método runInstances. O método retorna um objeto RunInstancesResult da seguinte forma:

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

Depois que a instância estiver em execução, você poderá se conectar a ela usando o par de chaves. Para obter mais informações, consulte Conectar-se à sua instância do Linux no Guia do usuário do Amazon EC2 para instâncias do Linux.

Usar perfis do IAM para conceder acesso a recursos da AWS no Amazon EC2

Todas as solicitações à Amazon Web Services (AWS) devem ser assinadas criptograficamente usando as credenciais emitidas pela AWS. Você pode usar perfis do IAM para conceder acesso seguro de maneira prática a recursos da AWS a partir das instâncias do Amazon EC2.

Este tópico fornece informações sobre como usar os perfis do IAM com aplicativos do Java SDK em execução no Amazon EC2. Para obter mais informações sobre instâncias do IAM, consulte Perfis do IAM para Amazon EC2 no Guia do usuário do Amazon EC2 para instâncias do Linux.

A cadeia de fornecedores padrão e os perfis de instância do EC2

Se o aplicativo criar um cliente da AWS usando o construtor padrão, o cliente vai procurar credenciais usando a cadeia de fornecedores de credenciais padrão, na seguinte ordem:

1. Nas propriedades do sistema Java: `aws.accessKeyId` e `aws.secretKey`.
2. Em variáveis de ambiente do sistema: `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY`.
3. No arquivo de credenciais padrão (o local desse arquivo varia de acordo com a plataforma).
4. Credenciais entregues por meio do serviço de contêiner do Amazon EC2 se a variável de ambiente `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` estiver definida e o gerente de segurança tiver permissão para acessar a variável.

5. Nas credenciais de perfil de instância, que existem dentro dos metadados da instância associados ao perfil do IAM para a instância do EC2.
6. Credenciais do token de identidade da web do ambiente ou contêiner.

A etapa credenciais de perfil de instância na cadeia de fornecedores padrão está disponível somente durante a execução do aplicativo em uma instância do Amazon EC2, mas proporciona a maior facilidade de uso e a melhor segurança durante o trabalho com instâncias do Amazon EC2. Você também pode passar uma instância InstanceProfileCredentialsProvider diretamente para o construtor cliente para obter as credenciais do perfil de instância sem avançar em toda a cadeia de fornecedores padrão.

Por exemplo:

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

Usando essa abordagem, o SDK recupera credenciais da AWS temporárias que tenham as mesmas permissões das associadas ao perfil do IAM ligada à instância do Amazon EC2 no perfil de instância. Embora essas credenciais sejam temporárias e acabem expirando, o `InstanceProfileCredentialsProvider` as atualiza periodicamente para você, de maneira que as credenciais obtidas continuem permitindo o acesso à AWS.

> **Important**
>
> A atualização de credenciais automática acontece somente quando você usa o construtor de cliente padrão, que cria o próprio `InstanceProfileCredentialsProvider` como parte da cadeia de fornecedores padrão, ou quando passa uma instância `InstanceProfileCredentialsProvider` diretamente para o construtor de cliente. Se usar outro método para obter ou passar credenciais de perfil de instância, você será responsável por verificar e atualizar as credenciais expiradas.

Se não conseguir encontrar credenciais usando a cadeia de fornecedores de credenciais, o construtor de cliente lançará um AmazonClientException.

Demonstração: usar funções do IAM em instâncias do EC2

A demonstração a seguir mostra como recuperar um objeto do Amazon S3 usando um perfil do IAM para gerenciar acesso.

Criar um perfil do IAM

Crie um perfil do IAM que conceda acesso somente leitura ao Amazon S3.

1. Abra o console do IAM.
2. No painel de navegação, selecione Roles e Create New Role.
3. Insira um nome para a função e selecione Next Step (Próxima etapa). Lembre-se desse nome, porque será necessário quando você executar a instância do Amazon EC2.
4. Na página Selecionar tipo de função, em Funções do AWS service (Serviço da AWS), selecione Amazon EC2.
5. Na página Definir permissões, em Selecionar modelo de política, selecione Acesso somente leitura do Amazon S3 e, em seguida, Próxima etapa.
6. Na página Review, selecione Create Role.

Iniciar uma instância do EC2 e especificar o perfil do IAM

Você pode iniciar uma instância do Amazon EC2 com um perfil do IAM usando o console do Amazon EC2 ou o AWS SDK for Java.

- Para ativar uma instância do Amazon EC2 usando o console, siga as orientações em Conceitos básicos das instâncias do Linux do Amazon EC2 no Guia do usuário do Amazon EC2 para instâncias do Linux.

  Quando você chegar à página Review Instance Launch (Revisar ativação da instância), selecione Edit instance details (Editar detalhes da instância). Em Perfil do IAM, escolha o perfil do IAM criado por você anteriormente. Conclua o procedimento conforme indicado.

  > **Note**
  >
  > Será necessário criar ou usar um grupo de segurança existente e um par de chaves para se conectar à instância.

- Para iniciar uma instância do Amazon EC2 com um perfil do IAM usando o AWS SDK for Java, consulte Executar uma instância do Amazon EC2.

Criar o aplicativo

Vamos criar o aplicativo de exemplo a ser executado na instância do EC2. Primeiro, crie um diretório que você possa usar para manter os arquivos de tutorial (por exemplo, `GetS3ObjectApp`).

Em seguida, copie as bibliotecas do AWS SDK for Java no diretório recém-criado. Se tiver obtido por download o AWS SDK for Java para o diretório `~/Downloads`, você poderá copiá-lo usando os seguintes comandos:

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

Abra um novo arquivo, chame-o de `GetS3Object.java` e adicione o seguinte código:

```
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();
  }
}
```

Abra um novo arquivo, chame-o de `build.xml` e adicione as seguintes linhas:

```
<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>
```

Compile e execute o programa modificado. Não há credenciais armazenadas no programa. Por isso, a menos que você tenha as credenciais da AWS já especificadas, o código lançará `AmazonServiceException`. Por exemplo:

```
$ 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
```

Transferir o programa compilado para a instância do EC2

Transfira o programa para a instância do Amazon EC2 usando uma cópia segura ( ), com as bibliotecas do AWS SDK for Java. A sequência de comandos é semelhante à sequência a seguir.

```
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**
>
> Dependendo da distribuição do Linux usada por você, o nome de usuário pode ser "ec2-user", "root" ou "ubuntu". Para obter o nome DNS público da instância, abra o console do EC2 e procure o valor Public DNS (DNS público) na guia Description (Descrição) (por exemplo, `ec2-198-51-100-1.compute-1.amazonaws.com`).

Nos comandos anteriores:

- `GetS3Object.class` é o programa compilado
- `build.xml` é o arquivo ant usado para compilar e executar o programa
- os diretórios `lib` e `third-party` são as pastas da biblioteca correspondente do AWS SDK for Java.

- A opção `-r` indica que `scp` deve fazer uma cópia recursiva de todo o conteúdo dos diretórios `library` e `third-party` na distribuição do AWS SDK for Java.
- A opção `-p` indica que `scp` deverá preservar as permissões dos arquivos de código-fonte quando copiá-los para o destino.

> **Note**
>
> A opção `-p` funciona somente no Linux, macOS ou Unix. Se estiver copiando arquivos do Windows, você precisará corrigir as permissões de arquivo na instância usando o seguinte comando:

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

Executar o programa de exemplo na instância do EC2

Para executar o programa, conecte-se à instância do Amazon EC2. Para obter mais informações, consulte Conectar-se à sua instância do Linux no Guia do usuário do Amazon EC2 para instâncias do Linux.

Se **`ant`** não estiver disponível na instância, instale-o usando o seguinte comando:

```
sudo yum install ant
```

Em seguida, execute o programa usando `ant` da seguinte maneira:

```
ant run
```

O programa gravará o conteúdo do objeto do Amazon S3 na janela de comando.

Tutorial: instâncias spot do Amazon EC2

Visão geral

As instâncias spot permitem que você faça ofertas pela capacidade não utilizada do Amazon Elastic Compute Cloud (Amazon EC2) em até 90% com relação ao preço de instância sob demanda e execute as instâncias adquiridas desde que seu lance exceda o Preço spot atual. O Amazon EC2 altera o preço spot periodicamente com base na oferta e na demanda e os clientes cujos

lances alcancem ou excedam o preço ganham acesso às instâncias spot disponíveis. Assim como instâncias sob demanda e instâncias reservadas, as instâncias spot são outra opção para obter mais capacidade computacional.

As instâncias spot podem reduzir significativamente os custos do Amazon EC2 para processamento em lote, pesquisa científica, processamento de imagens, codificação de vídeos, crawling de dados e web, análise financeira e testes. Além disso, as instâncias spot dão acesso a grandes quantidades de capacidade adicional em situações nas quais a necessidade dessa capacidade não é urgente.

Para usar instâncias spot, faça uma solicitação de instância spot especificando o preço máximo que você está disposto a pagar por hora de instância; esse é seu lance. Se seu lance exceder o preço spot atual, sua solicitação será cumprida e as instâncias serão executadas até que você opte por encerrá-las ou até que o preço spot exceda seu lance (o que ocorrer primeiro).

É importante observar:

- Você frequentemente pagará menos por hora que seu lance. O Amazon EC2 ajusta o preço spot periodicamente, à medida que as requisições entram e a oferta disponível se altera. Todos pagam o mesmo preço spot por esse período, independente de o lance ter sido maior. Portanto, você pode pagar menos que seu lance, mas jamais pagará mais.
- Se você estiver executando instâncias spot e o seu lance não atender mais ou ultrapassar o preço spot atual, suas instâncias serão encerradas. Isto significa que você precisará se certificar de que as suas cargas de trabalho e aplicativos sejam suficientemente flexíveis para se beneficiarem desta capacidade oportunista.

As instâncias spot funcionam exatamente como outras instâncias do Amazon EC2 durante a execução e, como outras instâncias do Amazon EC2, as instâncias spot poderão ser encerradas quando não forem mais necessárias. Se você finalizar sua instância, pagará por qualquer hora parcial usada (como para instâncias sob demanda ou reservadas). Entretanto, se o preço spot for além do lance e a instância for encerrada pelo Amazon EC2, você não será cobrado em relação a qualquer hora parcial de uso.

Este tutorial mostra como usar o AWS SDK for Java para fazer o seguinte.

- Enviar uma requisição spot
- Determinar quando a requisição spot é atendida
- Cancelar a requisição spot
- Encerrar as instâncias associadas

Pré-requisitos

Para usar este tutorial, você deve ter o AWS SDK for Java instalado, bem como ter atendido aos pré-requisitos de instalação básicos. Consulte Configurar o AWS SDK for Java para obter mais informações.

Etapa 1: configurar as credenciais

Para começar a usar esse exemplo de código, é preciso configurar credenciais da AWS. Consulte Configurar credenciais e a região da AWS para desenvolvimento para obter instruções sobre como fazer isso.

> **Note**
>
> Recomendamos usar as credenciais de um usuário do IAM para fornecer esses valores. Para obter mais informações, consulte Cadastrar-se na AWS e criar um usuário do IAM.

Agora que definiu as configurações, você pode começar a usar o código no exemplo.

Etapa 2: configurar um security group

Um security group funciona como um firewall que controla o tráfego permitido de entrada e saída de um grupo de instâncias. Por padrão, uma instância é iniciada sem nenhum security group, o que significa que todo o tráfego IP recebido, em qualquer porta TCP, será negado. Por isso, antes de enviar a requisição spot, vamos configurar um security group que permite o tráfego de rede necessário. Para os fins deste tutorial, criaremos um novo security group chamado "GettingStarted" que permite o tráfego Secure Shell (SSH) do endereço IP em que está executando o aplicativo. Para configurar um novo security group, você precisa incluir ou executar o exemplo de código a seguir que configura o security group de maneira programática.

Depois que criarmos um objeto de cliente `AmazonEC2`, criaremos um objeto `CreateSecurityGroupRequest` com o nome, "GettingStarted", e uma descrição do security group. Em seguida, chamaremos a API `ec2.createSecurityGroup` para criar o grupo.

Para permitir acesso ao grupo, criaremos um objeto `ipPermission` com o intervalo de endereços IP definido como a representação CIDR da sub-rede para o computador local; o sufixo "/10" no endereço IP indica a sub-rede do endereço IP especificado. Também configuramos o objeto `ipPermission` com o protocolo TCP e a porta 22 (SSH). A etapa final é chamar

`ec2.authorizeSecurityGroupIngress` com o nome do security group e o objeto `ipPermission`.

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

Você precisa somente executar esse aplicativo uma vez para criar um novo security group.

Você também pode criar o security group usando o AWS Toolkit for Eclipse. Consulte Gerenciar grupos de segurança do AWS Cost Explorer para obter mais informações.

Etapa 3: enviar a requisição spot

Para enviar uma requisição spot, é preciso primeiro determinar o tipo de instância, a imagem de máquina da Amazon (AMI) e o preço máximo do lance que você deseja usar. Você também deve incluir o security group que configuramos anteriormente, de maneira que possa fazer logon na instância, se desejado.

Há vários tipos de instância a escolher; acesse Tipos de instância do Amazon EC2 para ver a lista completa. Para este tutorial, usaremos t1.micro, o tipo de instância mais barata disponível. Em seguida, determinaremos o tipo de AMI que gostaríamos de usar. Usaremos ami-a9d09ed1, a AMI do Amazon Linux mais atualizada disponível quando elaboramos este tutorial. A AMI mais recente pode mudar ao longo do tempo, mas você pode sempre determinar a AMI da versão mais recente seguindo estas etapas:

1. Abra o console de Amazon EC2.
2. Selecione o botão Launch Instance (Iniciar instância).
3. A primeira janela exibe as AMIs disponíveis. O ID da AMI é exibido ao lado de cada título de AMI. Você também pode usar a API `DescribeImages`, mas aproveitar esse comando está fora do escopo deste tutorial.

Existem muitas maneiras de abordar lances para instâncias spot. Para obter uma visão geral ampla das diversas abordagens, assista ao vídeo Bidding for Spot Instances. No entanto, para começar, descreveremos três estratégias comuns: lance para garantir que o custo seja menor que a

definição de preço sob demanda, lance baseado no valor do cálculo resultante e lance para adquirir capacidade computacional o mais rápido possível.

- Reduzir custo abaixo da demanda Você tem um job de processamento em lote que modera determinado número de horas ou dias para ser executado. Contudo, você tem flexibilidade em relação ao início e ao fim quando terminar. Você quer ver se pode o concluído por um custo inferior ao das instâncias sob demanda. Você examina o histórico de preços spot para tipos de instância usando o Console de gerenciamento da AWS ou a API do Amazon EC2. Para obter mais informações, acesse Exibir histórico de preços spot. Depois de analisar o histórico de preços do tipo de instância desejado em determinada zona de disponibilidade, há duas abordagens alternativas para seu lance:
  - Você pode oferecer um lance na extremidade superior do intervalo de preços spot (que ainda estão abaixo do preço sob demanda), prevendo que sua solicitação spot única provavelmente seria cumprida e executada pelo tempo de computação consecutivo suficiente para concluir o trabalho.
  - Ou você pode especificar o valor que está disposto a pagar pelas instâncias spot como uma porcentagem do preço das instâncias sob demanda e planejar combinar muitas instâncias executadas ao longo do tempo por meio de uma requisição persistente. Se o preço especificado for excedido, a instância spot será encerrada. (Explicaremos como automatizar essa tarefa ainda neste tutorial.)
- Não pagar a mais pelo valor do resultado Você tem um trabalho de processamento de dados a ser executado. Você entende o valor dos resultados do trabalho bem o suficiente para saber o quanto valem em termos de custos computacionais. Após analisar o histórico de preços spot para seu tipo de instância, escolha o preço de lance no qual o custo do tempo computacional não é mais que o valor dos resultados do trabalho. Você cria um lance persistente e o deixa ser executado de forma intermitente, à medida que o preço spot flutua acima ou abaixo do seu lance.
- Adquirir capacidade computacional rapidamente Você tem a necessidade imprevista e de curto prazo por capacidade adicional indisponível pelas instâncias sob demanda. Depois de analisar o histórico de preços spot para seu tipo de instância, faça um lance acima do preço histórico mais alto para indicar uma maior probabilidade de sua solicitação ser atendida com rapidez e continuar a computação até a conclusão.

Depois de escolher o preço do lance, você já estará pronto para solicitar uma instância spot. Para os fins deste tutorial, daremos uma sugestão de preço sob demanda (USD 0,03) para maximizar as chances de o lance ser cumprido. Você pode determinar os tipos de instâncias disponíveis e os preços sob demanda para instâncias indo à página de definição de preços do Amazon EC2.

Enquanto as instâncias spot estão em execução, você paga o preço spot em vigor pelo período da execução das instâncias. Os preços de instância spot são definidos pelo Amazon EC2 e são ajustados gradualmente de acordo com tendências de longo prazo da oferta e da demanda pela capacidade de instâncias spot. Também é possível especificar o valor que você está disposto a pagar por uma instância spot como uma porcentagem do preço de instância sob demanda. Para solicitar uma instância spot, basta criar sua requisição com os parâmetros escolhidos anteriormente. Começamos criando um objeto `RequestSpotInstanceRequest`. O objeto de solicitação exige o número de instâncias que você deseja para começar e o preço do lance. Além disso, você precisa definir o `LaunchSpecification` para a solicitação, que inclui o tipo de instância, o ID do AMI e o security group que deseja usar. Depois que a solicitação for preenchida, você chamará o método `requestSpotInstances` no objeto `AmazonEC2Client`. O exemplo a seguir mostra como solicitar uma instância 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);
```

Executar esse código iniciará uma nova solicitação de instância spot. Há outras opções que você pode usar para configurar suas solicitações spot. Para saber mais, visite Tutorial: gerenciamento de requisições spot do Amazon EC2 avançado ou a classe RequestSpotInstances na Referência de API do AWS SDK for Java.

> **Note**
>
> Você será cobrado por todas as instâncias spot que forem efetivamente iniciadas, por isso cancele as solicitações e encerre todas as instâncias que iniciar para reduzir os encargos associados.

Etapa 4: determinar o estado da solicitação spot

Em seguida, queremos criar um código para esperar até que a solicitação spot alcance o estado "ativo" antes de continuar para a última etapa. Para determinar o estado da solicitação spot, faremos uma sondagem do método describeSpotInstanceRequests para obter o estado do ID da solicitação spot que queremos monitorar.

O ID da solicitação criado na Etapa 2 é integrado na resposta à solicitação `requestSpotInstances`. O código de exemplo a seguir mostra como coletar IDs de solicitação da resposta `requestSpotInstances` e usá-los para preencher um `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());
}
```

Para monitorar o ID da solicitação, chame o método `describeSpotInstanceRequests` para determinar o estado da solicitação. Em seguida, mantenha em loop até que a solicitação não esteja no estado "aberto". Monitoramos um estado de não "aberto", em vez de um estado de, digamos, "ativo", porque a solicitação poderá ir diretamente para "fechado" se houver um problema com os argumentos da solicitação. O código de exemplo a seguir apresenta os detalhes de como realizar essa tarefa.

```
// 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);
```

Depois de executar esse código, a solicitação da instância spot terá sido concluída ou falhado com um erro que será produzido para a tela. Em ambos os casos, podemos avançar à próxima etapa para limpar todas as solicitações ativas e encerrar as instâncias em execução.

Etapa 5: limpar as solicitações spot e instâncias

Por fim, precisamos limpar as solicitações e as instâncias. É importante cancelar todas as solicitações pendentes e encerrar as instâncias. Simplesmente cancelar as solicitações não encerrará as instâncias, o que significa que você continuará pagando por elas. Se você encerrar suas instâncias, suas solicitações spot poderão ser canceladas, mas há algumas situações, como se você usar lances persistentes, nas quais encerrar suas instâncias não basta para impedir que a solicitação seja cumprida novamente. Portanto, é uma prática recomendada cancelar todos os lances ativos e encerrar as usar instâncias em execução.

O código a seguir demonstra como cancelar as solicitações.

```
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());
}
```

Para encerrar todas as instâncias pendentes, você precisará do ID da instância associada à solicitação que as iniciou. O exemplo de código a seguir utiliza o código original para monitorar as instâncias e adiciona um `ArrayList` no qual armazenamos o ID da instância associado à resposta `describeInstance`.

```
// Create a variable that will track whether there are any requests
// still in the open state.
boolean anyOpen;
// Initialize variables.
ArrayList<String> instanceIds = new ArrayList<String>();

do {
   // Create the describeRequest with all of the request ids to
   // monitor (e.g. that we started).
   DescribeSpotInstanceRequestsRequest describeRequest = new
 DescribeSpotInstanceRequestsRequest();
   describeRequest.setSpotInstanceRequestIds(spotInstanceRequestIds);

   // Initialize the anyOpen variable to false, which assumes there
   // are no requests open unless we find one that is still open.
   anyOpen = false;

   try {
         // Retrieve all of the requests we want to monitor.
         DescribeSpotInstanceRequestsResult describeResult =
            ec2.describeSpotInstanceRequests(describeRequest);

         List<SpotInstanceRequest> describeResponses =
            describeResult.getSpotInstanceRequests();

         // Look through each request and determine if they are all
         // in the active state.
         for (SpotInstanceRequest describeResponse : describeResponses) {
           // If the state is open, it hasn't changed since we
           // attempted to request it. There is the potential for
           // it to transition almost immediately to closed or
           // cancelled so we compare against open instead of active.
           if (describeResponse.getState().equals("open")) {
              anyOpen = true; break;
           }
           // Add the instance id to the list we will
           // eventually terminate.
           instanceIds.add(describeResponse.getInstanceId());
```

```
         }
   } catch (AmazonServiceException e) {
      // If we have an exception, ensure we don't break out
      // of the loop. This prevents the scenario where there
      // was blip on the wire.
      anyOpen = true;
   }

    try {
        // Sleep for 60 seconds.
        Thread.sleep(60*1000);
    } catch (Exception e) {
        // Do nothing because it woke up early.
    }
} while (anyOpen);
```

Usando os IDs de instância, armazenados no `ArrayList`, encerre todas as instâncias em execução usando o trecho de código a seguir.

```
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());
}
```

Resumir

Para resumir, fornecemos uma abordagem mais orientada ao objeto que integra as etapas anteriores que mostramos: inicializar o EC2 Client, enviar a solicitação spot, determinar quando as solicitações spot não estão mais no estado aberto e limpar eventuais solicitações spot e instâncias associadas. Criamos uma classe chamada `Requests` que realiza essas ações.

Também criamos uma classe `GettingStartedApp`, que tem um método principal em que realizamos as chamadas à função de alto nível. Mais especificamente, inicializaremos o objeto

`Requests` descrito anteriormente. Enviaremos a solicitação da instância spot. Em seguida, aguarde a solicitação spot chegar ao estado "ativo". Por fim, limpamos as solicitações e as instâncias.

O código-fonte completo desse exemplo pode ser visualizado ou obtido por download no GitHub.

Parabéns! Você acabou de concluir o tutorial de conceitos básicos para desenvolver software de instância spot com o AWS SDK for Java.

Próximas etapas

Avance para Tutorial: gerenciamento de solicitações spot do Amazon EC2 avançado.

Tutorial: gerenciamento de requisições spot do Amazon EC2 avançado

Instâncias spot do Amazon EC2 permitem que você ofereça a capacidade não utilizada do Amazon EC2 e execute essas instâncias desde que a oferta ultrapasse o preço spot atual. O Amazon EC2 altera o preço spot periodicamente com base na oferta e na demanda. Para obter mais informações sobre instâncias spot, consulte Instâncias spot no Guia do usuário do Amazon EC2 para instâncias do Linux.

Pré-requisitos

Para usar este tutorial, você deve ter o AWS SDK for Java instalado, bem como ter atendido aos pré-requisitos de instalação básicos. Consulte Configurar o AWS SDK for Java para obter mais informações.

Configurar as credenciais

Para começar a usar esse exemplo de código, é preciso configurar credenciais da AWS. Consulte Configurar credenciais e a região da AWS para desenvolvimento para obter instruções sobre como fazer isso.

> **Note**
>
> Recomendamos usar as credenciais de um usuário do IAM para fornecer esses valores. Para obter mais informações, consulte Cadastrar-se na AWS e criar um usuário do IAM.

Agora que definiu as configurações, você pode começar a usar o código no exemplo.

Configurar um security group

Um security group funciona como um firewall que controla o tráfego permitido de entrada e saída de um grupo de instâncias. Por padrão, uma instância é iniciada sem nenhum security group, o que significa que todo o tráfego IP recebido, em qualquer porta TCP, será negado. Por isso, antes de enviar a requisição spot, vamos configurar um security group que permite o tráfego de rede necessário. Para os fins deste tutorial, criaremos um novo security group chamado "GettingStarted" que permite o tráfego Secure Shell (SSH) do endereço IP em que está executando o aplicativo. Para configurar um novo security group, você precisa incluir ou executar o exemplo de código a seguir que configura o security group de maneira programática.

Depois que criarmos um objeto de cliente `AmazonEC2`, criaremos um objeto `CreateSecurityGroupRequest` com o nome, "GettingStarted", e uma descrição do security group. Em seguida, chamaremos a API `ec2.createSecurityGroup` para criar o grupo.

Para permitir acesso ao grupo, criaremos um objeto `ipPermission` com o intervalo de endereços IP definido como a representação CIDR da sub-rede para o computador local; o sufixo "/10" no endereço IP indica a sub-rede do endereço IP especificado. Também configuramos o objeto `ipPermission` com o protocolo TCP e a porta 22 (SSH). A etapa final é chamar `ec2 .authorizeSecurityGroupIngress` com o nome do security group e o objeto `ipPermission`.

(O código a seguir é o mesmo que usamos no primeiro 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());
}
```

Você pode visualizar todo esse exemplo de código em `advanced.CreateSecurityGroupApp.java`. Você precisa somente executar esse aplicativo uma vez para criar um novo security group.

> **Note**
>
> Você também pode criar o security group usando o AWS Toolkit for Eclipse. Consulte Gerenciar grupos de segurança do AWS Cost Explorer no Guia do usuário do AWS Toolkit for Eclipse para obter mais informações.

Opções de criação da requisição da instância spot detalhadas

Como explicamos no Tutorial: instâncias spot do Amazon EC2, você precisa compilar a requisição com um tipo de instância, uma Imagem de máquina da Amazon (AMI) e um preço de sugestão de preço máximo.

Vamos começar criando um objeto `RequestSpotInstanceRequest`. O objeto de requisição exige o número de instâncias que você deseja e o preço da sugestão. Além disso, precisamos definir o `LaunchSpecification` da requisição, que inclui o tipo de instância, o ID de AMI e o security group que você deseja usar. Depois que a requisição for preenchida, chamaremos o método `requestSpotInstances` no objeto `AmazonEC2Client`. Veja a seguir um exemplo de como solicitar uma instância spot.

(O código a seguir é o mesmo que usamos no primeiro 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);
```

Requisições persistentes x ocasionais

Ao compilar uma requisição spot, você pode especificar diversos parâmetros opcionais. O primeiro é se a requisição é somente ocasional ou persistente. Por padrão, trata-se de uma requisição ocasional. A requisição ocasional pode ser atendida somente uma vez e, depois que as instâncias solicitadas forem encerradas, a requisição será fechada. Uma requisição persistente é considerada para o cumprimento sempre que não há instância spot em execução para a mesma requisição. Para especificar o tipo de requisição, basta definir o tipo na requisição spot. Isso pode ser feito com o código a seguir.

```
// 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);
```

Limitar a duração de uma requisição

Você também pode especificar o tempo em que a requisição permanecerá válida. Você pode especificar horários de início e término para esse período. Por padrão, uma requisição spot será considerada para o cumprimento a partir do momento em que é criada até ser atendida ou cancelada por você. No entanto, você poderá restringir o período de validade, se precisar. Um exemplo de como especificar esse período é mostrado no código a seguir.

```
// 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);
```

Agrupar as requisições de instância spot do Amazon EC2

Você tem a opção de agrupar as requisições de instância spot de diversas maneiras diferentes. Veremos os benefícios de usar grupos de inicialização, grupos de zonas de disponibilidade e grupos de colocações.

Se quiser garantir que as instâncias spot sejam todas executadas e encerradas juntas, você terá a opção de aproveitar um grupo de inicialização. Grupo de inicialização é um rótulo que agrupa um conjunto de sugestões de preço. Todas as instâncias em um grupo de execução são iniciadas e encerradas juntas. Se instâncias em um grupo de inicialização já tiverem sido atendidas, não haverá garantia de que novas instâncias executadas com o mesmo grupo de inicialização também serão

atendidas. Um exemplo de como definir um grupo de inicialização é mostrado no exemplo de código a seguir.

```
// 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 quiser garantir que todas as instâncias dentro de uma solicitação sejam iniciadas na mesma zona de disponibilidade e não se preocupar com qual delas, será possível aproveitar os grupos de zonas de disponibilidade. Grupo de zonas de disponibilidade é um rótulo que agrupa um conjunto de instâncias na mesma zona de disponibilidade. Todas as instâncias que compartilham um grupo de zonas de disponibilidade e são atendidas simultaneamente começarão na mesma zona de disponibilidade. Um exemplo de como definir um grupo de zonas de disponibilidade está a seguir.

```
// 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);
```

Você pode especificar uma zona de disponibilidade desejada para as instâncias spot. O código de exemplo a seguir mostra como definir uma zona de disponibilidade.

```
// 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);
```

Por fim, será possível especificar um grupo de colocações se estiver usando instâncias spot High Performance Computing (HPC – Computação de Alto Desempenho), como instâncias de computação em cluster ou de GPU de cluster. Os grupos de colocações oferecem latência mais baixa e alta conectividade de largura de banda entre as instâncias. Um exemplo de como definir um grupo de colocações está a seguir.

```
// 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);
```

Todos os parâmetros mostrados nesta seção são opcionais. Também é importante perceber que a maioria desses parâmetros (com exceção da sugestão de preço ser ocasional ou persistente) pode reduzir a probabilidade de cumprimento da sugestão de preço. Por isso, será importante aproveitar essas opções somente se você precisar delas. Todos os exemplos de código anteriores são integrados a um exemplo longo, que pode ser encontrado na classe `com.amazonaws.codesamples.advanced.InlineGettingStartedCodeSampleApp.java`.

Como manter uma partição raiz após a interrupção ou o encerramento

Uma das maneiras mais fáceis de gerenciar a interrupção das instâncias spot é garantir que os dados sejam verificados em um volume do Amazon Elastic Block Store (Amazon Amazon EBS) em um ritmo regular. Verificando periodicamente, se houver uma interrupção, você perderá somente os dados criados desde o ponto de verificação mais recente (pressupondo-se que não haja outras

ações não idempotentes realizadas entre essas duas situações). Para facilitar ainda mais esse processo, você pode configurar a requisição spot para garantir que a partição raiz não seja excluída na interrupção ou no encerramento. Inserimos um novo código no exemplo a seguir que mostra como habilitar esse cenário.

No código adicionado, criamos um objeto `BlockDeviceMapping` e definimos o Amazon Elastic Block Store (Amazon EBS) associado como um objeto Amazon EBS que configuramos como `not` caso a instância spot seja encerrada. Em seguida, adicionaremos o `BlockDeviceMapping` a ArrayList de mapeamentos que incluímos na especificação de inicialização.

```
// 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);
```

Pressupondo-se que queira reanexar esse volume à instância na inicialização, você também pode usar as configurações de mapeamento de dispositivos de blocos. Como alternativa, se você tiver anexado uma partição não raiz, será possível especificar os volumes do Amazon Amazon EBS que queira anexar à instância spot após o reinício. Você pode fazer isso simplesmente especificando um ID de snapshot no `EbsBlockDevice` e um nome de dispositivo alternativo nos objetos `BlockDeviceMapping`. Utilizando-se mapeamentos de dispositivos de blocos, pode ser mais fácil inicializar a instância.

Usar a partição raiz no ponto de verificação dos dados críticos é uma ótima maneira de gerenciar o potencial de interrupção das instâncias. Para obter mais métodos para gerenciar o potencial de interrupção, assista ao vídeo Gerenciar interrupções.

Como marcar as requisições spot e as instâncias

Adicionar tags a recursos do Amazon EC2 pode simplificar a administração da infraestrutura em nuvem. Uma forma de metadados, as tags podem ser usadas para criar nomes amigáveis ao

usuário, aprimorar a capacidade de pesquisa e melhorar a coordenação entre vários usuários. Você também pode usar tags para automatizar scripts e partes dos processos. Para ler mais sobre como marcar recursos do Amazon EC2, acesse Usar tags no Guia do usuário do Amazon EC2 para instâncias do Linux.

Marcar requisições

Para adicionar tags às requisições spot, você precisará marcá-las depois que tiverem sido solicitadas. O valor de retorno de `requestSpotInstances()` fornece um objeto RequestSpotInstancesResult que você pode usar para obter os IDs de requisição spot para marcar:

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

Assim que você tiver os IDs, será possível marcar as solicitações adicionando os IDs a um CreateTagsRequest e chamando o método `createTags()` do cliente do Amazon EC2:

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

Marcar instâncias

De maneira semelhante a requisições spot, você poderá marcar somente uma instância depois que ela tiver sido criada, o que acontecerá assim que a requisição spot tiver sido atendida (deixa de estar no estado aberto).

É possível verificar o status das solicitações chamando o método `describeSpotInstanceRequests()` do cliente do Amazon EC2 com um objeto DescribeSpotInstanceRequestsRequest. O objeto DescribeSpotInstanceRequestsResult retornado contém uma lista de objetos SpotInstanceRequest que você pode usar para consultar o status das requisições spot e obter os IDs de instância assim que elas não estiverem mais no estado aberto.

Assim que a requisição spot não estiver mais aberta, você poderá recuperar o ID de instância do objeto `SpotInstanceRequest` chamando o método `getInstanceId()`.

```
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);
```

Você já pode marcar as instâncias retornadas:

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

Cancelar requisições spot e encerrar instâncias

Cancelar uma requisição spot

Para cancelar uma requisição de instância spot, chame `cancelSpotInstanceRequests` no cliente do Amazon EC2 com um objeto CancelSpotInstanceRequestsRequest.

```
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());
}
```

Encerrar instâncias spot

É possível encerrar todas as instâncias spot em execução passando os IDs para o método `terminateInstances()` do cliente do Amazon EC2.

```
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());
}
```

Resumir

Para resumir, fornecemos uma abordagem mais orientada a objeto que integra as etapas que mostramos neste tutorial em uma única classe fácil de usar. Instanciamos uma classe chamada `Requests` que realiza essas ações. Também criamos uma classe `GettingStartedApp`, que tem um método principal em que realizamos as chamadas à função de alto nível.

O código-fonte completo desse exemplo pode ser visualizado ou obtido por download no GitHub.

Parabéns! Você concluiu o tutorial Recursos de solicitação avançados para desenvolver o software de instância spot com o AWS SDK for Java.

Gerenciar instâncias do Amazon EC2

Criar uma instância

Crie uma instância do Amazon EC2 chamando o método `runInstances` do AmazonEC2Client, fornecendo um RunInstancesRequest contendo a imagem de máquina da Amazon (AMI) a ser usada e um tipo de instância.

Importações

```
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;
```

Código da

```
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();
```

Consulte o exemplo completo.

Iniciar uma instância

Para iniciar uma instância do Amazon EC2, chame o método `startInstances` do AmazonEC2Client, fornecendo um StartInstancesRequest contendo o ID da instância para iniciar.

Importações

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

Código da

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

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

ec2.startInstances(request);
```

Consulte o exemplo completo.

Interromper uma instância

Para interromper uma instância do Amazon EC2, chame o método `stopInstances` do AmazonEC2Client, fornecendo um StopInstancesRequest contendo o ID da instância para interromper.

Importações

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

Código da

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

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

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

Consulte o exemplo completo.

Como reinicializar uma instância

Para reinicializar uma instância do Amazon EC2, chame o método `rebootInstances` do AmazonEC2Client, fornecendo um RebootInstancesRequest contendo o ID da instância para reinicializar.

Importações

```
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;
```

Código da

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

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

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

Consulte o exemplo completo.

Descrever instâncias

Para listar as instâncias, crie um DescribeInstancesRequest e chame o método `describeInstances` do AmazonEC2Client. Isso retornará um objeto DescribeInstancesResult que você pode usar para listar as instâncias do Amazon EC2 para a conta e a região.

As instâncias são agrupadas por reserva. Cada reserva corresponde à chamada a `startInstances` que iniciou a instância. Para listar as instâncias, você deve primeiro chamar a classe `DescribeInstancesResult` `getReservations' method, and then call` `getInstances` em cada objeto Reservation retornado.

Importações

```
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;
```

Código da

```
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;
    }
}
```

Os resultados são paginados. É possível obter mais resultados passando o valor retornado do método `getNextToken` do objeto resultante para o método `setNextToken` do objeto de solicitação original e usando o mesmo objeto de solicitação na próxima chamada para `describeInstances`.

Consulte o exemplo completo.

Monitorar uma instância

Você pode monitorar diversos aspectos das instâncias do Amazon EC2, como utilização de CPU e rede, memória disponível e espaço em disco restante. Para saber mais sobre monitoramento de instância, consulte Monitoramento do Amazon EC2 no Guia do Usuário do Amazon EC2 para instâncias do Linux.

Para iniciar o monitoramento de uma instância, você deve criar um MonitorInstancesRequest com o ID da instância para monitorar e passá-lo para o método `monitorInstances` do AmazonEC2Client.

Importações

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

Código da

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

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

ec2.monitorInstances(request);
```

Consulte o exemplo completo.

Interromper monitoramento de instâncias

Para interromper o monitoramento de uma instância, crie um UnmonitorInstancesRequest com o ID da instância para interromper o monitoramento e passá-lo para o método `unmonitorInstances` do AmazonEC2Client.

Importações

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

Código da

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

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

ec2.unmonitorInstances(request);
```

Consulte o exemplo completo.

Mais informações

- RunInstances na Referência de API do Amazon EC2
- DescribeInstances na Referência de API do Amazon EC2
- StartInstances na Referência de API do Amazon EC2
- StopInstances na Referência de API do Amazon EC2
- RebootInstances na Referência de API do Amazon EC2
- MonitorInstances na referência de API do Amazon EC2
- UnmonitorInstances na referência de API do Amazon EC2

Usar endereços IP elásticos no Amazon EC2

O EC2-Classic será descontinuado

> **Warning**
>
> Estamos aposentando o EC2-Classic em 15 de agosto de 2022. É recomendável migrar do EC2-Classic para uma VPC. Consulte mais informações na publicação de blog EC2-Classic Networking is Retiring – Here's How to Prepare.

Como alocar um endereço IP elástico

Para usar um endereço IP elástico, você primeiro aloca um para sua conta e o associa à instância ou a uma interface de rede.

Para alocar um endereço IP elástico, chame o método `allocateAddress` do AmazonEC2Client com um objeto AllocateAddressRequest contendo o tipo de rede (EC2 ou VPC clássico).

O AllocateAddressResult retornado contém um ID de alocação que é possível usar para associar o endereço a uma instância, passando o ID de alocação e o ID de instância em um AssociateAddressRequest para o método `associateAddress` do AmazonEC2Client.

Importações

```
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;
```

Código da

```
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);
```

Consulte o exemplo completo.

Descrever endereços IP elásticos

Para listar os endereços IP elásticos atribuídos à conta, chame o método `describeAddresses` do AmazonEC2Client. Ele retorna um DescribeAddressesResult que você pode usar para obter uma lista de objetos Address que representam os endereços IP elásticos na conta.

Importações

```
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;
```

Código da

```
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());
}
```

Consulte o exemplo completo.

Como liberar um endereço IP elástico

Para liberar um endereço IP elástico, chame o método `releaseAddress` do AmazonEC2Client, passando um ReleaseAddressRequest contendo o ID de alocação do endereço IP elástico que você deseja liberar.

Importações

```
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;
```

Código da

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

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

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

Depois que você liberar um endereço IP elástico, ele será liberado para o grupo de endereços IP da AWS e poderá estar indisponível em seguida. Não se esqueça de atualizar os registros DNS e todos os servidores ou dispositivos que se comunicam com o endereço. Se tentar liberar um endereço IP elástico já liberado, você receberá um erro AuthFailure se o endereço já estiver alocado para outra Conta da AWS.

Se você estiver usando o EC2-Classic ou uma VPC padrão, liberar um endereço IP elástico o desassociará automaticamente de qualquer instância a qual esteja associada. Para desassociar um endereço IP elástico sem liberá-lo, use o método `disassociateAddress` do AmazonEC2Client.

Se estiver usando uma VPC não padrão, você deverá usar `disassociateAddress` para desassociar o endereço IP elástico antes de tentar liberá-lo. Do contrário, o Amazon EC2 retornará um erro (InvalidIPAddress.InUse).

Consulte o exemplo completo.

Mais informações

- Endereços IP elásticos no Guia do Usuário do Amazon EC2 para instâncias Linux
- AllocateAddress na Referência de API do Amazon EC2
- DescribeAddresses na Referência de API do Amazon EC2
- ReleaseAddress na Referência de API do Amazon EC2

Usar regiões e zonas de disponibilidade

Descrever regiões

Para listar as regiões disponíveis para a conta, chame o método `describeRegions` do AmazonEC2Client. Ele retorna um DescribeRegionsResult. Chame o método `getRegions` do objeto retornado para obter uma lista de objetos Region que representam cada região.

Importações

```
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;
```

Código da

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

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

Consulte o exemplo completo.

Descrever zonas de disponibilidade

Para listar cada zona de disponibilidade disponível para a conta, chame o método `describeAvailabilityZones` do AmazonEC2Client. Ele retorna um DescribeAvailabilityZonesResult. Chame o método `getAvailabilityZones` para obter uma lista de objetos AvailabilityZone que representam cada zona de disponibilidade.

Importações

```
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;
```

Código da

```
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());
}
```

Consulte o exemplo completo.

Descrever contas

Para descrever a conta, chame o método `describeAccountAttributes` do AmazonEC2Client. Esse método retorna um objeto DescribeAccountAttributesResult. Invoque o método `getAccountAttributes` desses objetos para obter uma lista de objetos AccountAttribute. É possível percorrer a lista para recuperar um objeto AccountAttribute.

Você pode obter os valores dos atributos da sua conta invocando o método `getAttributeValues` do objeto AccountAttribute. Esse método retorna uma lista de objetos AccountAttributeValue. É possível percorrer essa segunda lista para exibir o valor dos atributos (veja o exemplo de código a seguir).

Importações

```
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;
```

Código da

```
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();
}
```

Veja o exemplo completo no GitHub.

Mais informações

- Regiões e zonas de disponibilidade no Guia do Usuário do Amazon EC2 para Instâncias do Linux
- DescribeRegions na Referência de API do Amazon EC2
- DescribeAvailabilityZones na Referência de API do Amazon EC2

Trabalhar com pares de chaves do Amazon EC2

Criação de um par de chaves

Para criar um par de chaves, chame o método `createKeyPair` do AmazonEC2Client com um CreateKeyPairRequest que contenha o nome da chave.

Importações

```
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;
```

Código da

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

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

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

Consulte o exemplo completo.

Descrever pares de chaves

Para listar os pares de chaves ou obter informações sobre eles, chame o método `describeKeyPairs` do AmazonEC2Client. Ele retorna um DescribeKeyPairsResult que você pode usar para acessar a lista de pares de chaves chamando o método `getKeyPairs`, que retorna uma lista de objetos KeyPairInfo.

Importações

```
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;
```

Código da

```
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());
}
```

Consulte o exemplo completo.

Excluir um par de chaves

Para excluir um par de chaves, chame o método `deleteKeyPair` do AmazonEC2Client, passando um DeleteKeyPairRequest que contenha o nome do par de chaves a ser excluído.

Importações

```
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;
```

Código da

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

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

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

Consulte o exemplo completo.

Mais informações

- Pares de chaves do Amazon EC2 no Guia do Usuário do Amazon EC2 para instâncias do Linux
- CreateKeyPair na Referência de API do Amazon EC2
- DescribeKeyPairs na Referência de API do Amazon EC2
- DeleteKeyPair na Referência de API do Amazon EC2

Como trabalhar com grupos de segurança no Amazon EC2

Criar um grupo de segurança

Para criar um grupo de segurança, chame o método `createSecurityGroup` do AmazonEC2Client com um CreateSecurityGroupRequest que contenha o nome da chave.

Importações

```
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;
```

Código da

```
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);
```

Consulte o exemplo completo.

Configurar um grupo de segurança

Um grupo de segurança pode controlar os tráfegos de entrada e saída para as instâncias do Amazon EC2.

Para adicionar regras de entrada ao grupo de segurança, use o método `authorizeSecurityGroupIngress` do AmazonEC2Client, fornecendo o nome do grupo de segurança e as regras de acesso (IpPermission) que você deseja atribuir a ele dentro de um objeto AuthorizeSecurityGroupIngressRequest. O exemplo a seguir mostra como adicionar permissões de IP a um grupo de segurança.

Importações

```
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;
```

Código da

```
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);
```

Para adicionar uma regra de saída ao grupo de segurança, forneça dados semelhantes em um AuthorizeSecurityGroupEgressRequest ao método `authorizeSecurityGroupEgress` do AmazonEC2Client.

Consulte o exemplo completo.

Descrever grupos de segurança

Para descrever os grupos de segurança ou obter informações sobre eles, chame o método `describeSecurityGroups` do AmazonEC2Client. Ele retorna um DescribeSecurityGroupsResult que você pode usar para acessar a lista de grupos de segurança chamando o método `getSecurityGroups`, que retorna uma lista de objetos SecurityGroup.

Importações

```
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;
```

Código da

```
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];
```

Consulte o exemplo completo.

Excluir um grupo de segurança

Para excluir um grupo de segurança, chame o método `deleteSecurityGroup` do AmazonEC2Client, passando um DeleteSecurityGroupRequest que contenha o ID do grupo de segurança a ser excluído.

Importações

```
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;
```

Código da

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

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

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

Consulte o exemplo completo.

Mais informações

- Grupos de segurança do Amazon EC2 no Guia do Usuário Amazon EC2 para Instâncias do Linux
- Autorização de tráfego de entrada para suas instâncias Linux no Guia do Usuário Amazon EC2 para Instâncias do Linux
- CreateSecurityGroup, na Referência de API do Amazon EC2
- DescribeSecurityGroups, na Referência de API do Amazon EC2
- DeleteSecurityGroup na Referência de API do Amazon EC2
- AuthorizeSecurityGroupIngress na Referência de API do Amazon EC2

Exemplos do IAM usando o AWS SDK for Java

Esta seção fornece exemplos de como programar o IAM usando o AWS SDK for Java.

O AWS Identity and Access Management (IAM) permite que você controle com segurança o acesso aos serviços e recursos da AWS para seus usuários. Usando o IAM, você pode criar e gerenciar usuários e grupos da AWS e usar permissões para permitir e negar o acesso deles aos recursos da AWS. Para obter um guia completo do IAM, visite o Guia do usuário do IAM.

> **Note**
>
> Entre os exemplos está somente o código necessário para demonstrar cada técnica. O código de exemplo completo está disponível no GitHub. A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

- Gerenciar chaves de acesso do IAM
- Gerenciar usuários do IAM
- Usar aliases de conta do IAM
- Trabalhar com políticas do IAM
- Trabalhar com certificados de servidor do IAM

Gerenciar chaves de acesso do IAM

Criar uma chave de acesso

Para criar uma chave de acesso do IAM, chame o método `createAccessKey` do AmazonIdentityManagementClient com um objeto CreateAccessKeyRequest.

`CreateAccessKeyRequest` tem dois construtores: um que utiliza um nome de usuário e outro sem parâmetros. Se usar a versão que não utiliza parâmetros, você deverá definir o nome de usuário usando o método setter `withUserName` para passá-lo ao método `createAccessKey`.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Listar chave de acesso

Para listar as chaves de acesso de um determinado usuário, crie um objeto ListAccessKeysRequest que contenha o nome de usuário cujas chaves listar e passe para o método `listAccessKeys` do AmazonIdentityManagementClient.

> **Note**
>
> Se você não fornecer um nome de usuário para `listAccessKeys`, ele tentará listar chaves de acesso associadas à Conta da AWS que assinou a solicitação.

Importações

```
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;
```

Código da

```
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;
    }
}
```

Os resultados de `listAccessKeys` são paginados (com um máximo de 100 registros por chamada). Você pode chamar `getIsTruncated` no objeto ListAccessKeysResult retornado para ver se a consulta retornou menos resultados então disponíveis. Dessa forma, chame `setMarker` no `ListAccessKeysRequest` e repasse para a próxima invocação de `listAccessKeys`.

Veja o exemplo completo no GitHub.

Recuperar a hora do uso mais recente de uma chave de acesso

Para obter a hora em que uma chave de acesso foi usada pela última vez, chame o método `getAccessKeyLastUsed` do AmazonIdentityManagementClient com o ID da chave de acesso, que pode ser passado usando um objeto GetAccessKeyLastUsedRequest ou diretamente para a sobrecarga que utiliza o ID de chave de acesso diretamente.

Depois disso, é possível usar o objeto GetAccessKeyLastUsedResult retornado para recuperar a hora em que a chave foi usada mais recentemente.

Importações

```
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;
```

Código da

```
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());
```

Veja o exemplo completo no GitHub.

Ativar ou desativar chaves de acesso

É possível ativar ou desativar uma chave de acesso criando um objeto UpdateAccessKeyRequest, fornecendo o ID de chave de acesso, como opção o nome do usuário e o Status desejado e passando o objeto de solicitação para o método `updateAccessKey` do AmazonIdentityManagementClient.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Excluir uma chave de acesso

Para excluir permanentemente uma chave de acesso, chame o método `deleteKey` do AmazonIdentityManagementClient, fornecendo um DeleteAccessKeyRequest que contenha o ID e o nome de usuário da chave de acesso.

> **Note**
>
> Depois de excluída, uma chave não poderá mais ser recuperada ou usada. Para desativar temporariamente uma chave de maneira que ela possa ser reativado mais tarde, use o método updateAccessKey em seu lugar.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Mais informações

- CreateAccessKey na Referência de API do IAM
- ListAccessKeys na Referência de API do IAM
- GetAccessKeyLastUsed na Referência de API do IAM
- UpdateAccessKey na Referência de API do IAM
- DeleteAccessKey na Referência de API do IAM

Gerenciar usuários do IAM

Criação de um usuário

Crie um usuário do IAM fornecendo o nome de usuário para o método `createUser` do AmazonIdentityManagementClient, diretamente ou usando um objeto CreateUserRequest que contém o nome do usuário.

Importações

```
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;
```

Código da

```
final AmazonIdentityManagement iam =
```

```
    AmazonIdentityManagementClientBuilder.defaultClient();

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

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

Veja o exemplo completo no GitHub.

Listar usuários

Para listar os usuários do IAM da conta, crie um ListUsersRequest e passe para o método `listUsers` do AmazonIdentityManagementClient. Você pode recuperar a lista de usuários chamando `getUsers` no objeto ListUsersResult retornado.

A lista de usuários retornados por `listUsers` é paginada. Você pode verificar se há mais resultados a serem recuperados chamando o método `getIsTruncated` do objeto de resposta. Se ele retornar `true`, chame o método `setMarker()` do objeto da solicitação, passando o valor de retorno do método `getMarker()` do objeto de resposta.

Importações

```
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;
```

Código da

```
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;
    }
}
```

Veja o exemplo completo no GitHub.

Atualizar um usuário

Para atualizar um usuário, chame o método `updateUser` do objeto do AmazonIdentityManagementClient, que utiliza um objeto UpdateUserRequest que pode ser usado por você para alterar o nome ou o caminho do usuário.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Excluir um usuário

Para excluir um usuário, chame a solicitação `deleteUser` do AmazonIdentityManagementClient com um objeto UpdateUserRequest definido com o nome de usuário a ser excluído.

Importações

```
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;
```

Código da

```
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;
}
```

Veja o exemplo completo no GitHub.

Mais informações

- Usuários do IAM no Guia do usuário do IAM
- Gerenciar usuários do IAM no Guia do usuário do IAM
- CreateUser na Referência de API do IAM
- ListUsers na Referência de API do IAM
- UpdateUser na Referência de API do IAM
- DeleteUser na Referência de API do IAM

Usar aliases de conta do IAM

Se deseja que o URL para sua página de login contenha o nome da sua empresa (ou outro identificador amigável) em vez do ID da sua Conta da AWS, você pode criar um alias para o Conta da AWS.

> **Note**
>
> A AWS dá suporte a exatamente um alias por conta.

Criar um alias da conta

Para criar um alias de conta, chame o método `createAccountAlias` do AmazonIdentityManagementClient com um objeto CreateAccountAliasRequest que contém o nome de alias.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Listar aliases de conta

Para listar o alias da conta, se houver, chame o método `listAccountAliases` do AmazonIdentityManagementClient.

> **Note**
>
> O ListAccountAliasesResult retornado dá suporte aos mesmos métodos `getIsTruncated` e `getMarker` como outros métodos list do AWS SDK for Java, mas uma Conta da AWS pode ter somente um alias de conta.

importações

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

código

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

ListAccountAliasesResult response = iam.listAccountAliases();

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

Veja o exemplo completo no GitHub.

Excluir um alias de conta

Para excluir o alias da sua conta, chame o método `deleteAccountAlias` do AmazonIdentityManagementClient. Ao excluir um alias de conta, você deve fornecer o nome usando um objeto DeleteAccountAliasRequest.

importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Mais informações

- ID da sua conta da AWS e o alias no Guia do Usuário do IAM
- CreateAccountAlias na Referência de API do IAM
- ListAccountAliases na Referência de API do IAM
- DeleteAccountAlias na Referência de API do IAM

Trabalhar com políticas do IAM

Criar uma política

Para criar uma política, forneça o nome da política e um documento de política em formato JSON em um CreatePolicyRequest para o método `createPolicy` do AmazonIdentityManagementClient.

Importações

```
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;
```

Código da

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

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

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

Os documentos de política do IAM; são strings JSON com uma sintaxe bem documentada. Veja a seguir um exemplo que fornece acesso para fazer solicitações específicas ao 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\"" +
    "    }" +
    "   ]" +
    "}";
```

Veja o exemplo completo no GitHub.

Obter uma política

Para recuperar uma política existente, chame o método `getPolicy` do AmazonIdentityManagementClient fornecendo o ARN da política em um objeto GetPolicyRequest.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Anexar uma política de função

Você pode anexar uma política a um perfil do IAM [http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html] chamando o método `attachRolePolicy` do AmazonIdentityManagementClient, fornecendo a ele o nome do perfil e o ARN da política em um AttachRolePolicyRequest.

Importações

```
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;
```

Código da

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

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

iam.attachRolePolicy(attach_request);
```

Veja o exemplo completo no GitHub.

Listar políticas de função anexadas

Liste as políticas anexadas em um perfil chamando o método `listAttachedRolePolicies` do AmazonIdentityManagementClient. Ele utiliza um objeto ListAttachedRolePoliciesRequest que contém o nome da função para listar as políticas.

Chame `getAttachedPolicies` no objeto ListAttachedRolePoliciesResult retornado para obter a lista de políticas anexadas. Os resultados podem ser truncados. Se o método `ListAttachedRolePoliciesResult` do objeto `getIsTruncated` retornar `true`, chame o método `ListAttachedRolePoliciesRequest` do objeto `setMarker` e o use para chamar `listAttachedRolePolicies` novamente a fim de obter o próximo lote de resultados.

Importações

```
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;
```

Código da

```
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());
}
```

Veja o exemplo completo no GitHub.

Desanexar uma política de função

Para desanexar uma política de uma função, chame o método `detachRolePolicy` do AmazonIdentityManagementClient fornecendo o nome da função e o ARN da política em um DetachRolePolicyRequest.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Mais informações

- Visão geral de políticas do IAM no Guia do usuário do IAM.
- Referência da política do IAM da AWS no Guia do usuário do IAM.
- CreatePolicy na Referência de API do IAM
- GetPolicy na Referência de API do IAM
- AttachRolePolicy na Referência de API do IAM
- ListAttachedRolePolicies na Referência de API do IAM
- DetachRolePolicy na Referência de API do IAM

Trabalhar com certificados de servidor do IAM

Para habilitar conexões HTTPS para o seu site ou aplicativo na AWS, você precisa de um certificado de servidor SSL/TLS. Você pode usar um certificado de servidor fornecido pelo AWS Certificate Manager ou um obtido junto a um provedor externo.

Recomendamos que você use o ACM para provisionar, gerenciar e implantar seus certificados de servidor. Com o ACM; você pode solicitar um certificado, implantá-lo nos seus recursos da AWS e deixar o ACM processar renovações de certificado para você. Os certificados fornecidos pelo ACM são gratuitos. Para obter mais informações sobre o ACM, consulte o Guia do usuário do ACM.

Obter um certificado de servidor

Você pode recuperar um certificado de servidor chamando o método `getServerCertificate` do AmazonIdentityManagementClient, passando um GetServerCertificateRequest com o nome do certificado.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Listar certificados de servidor

Para listar os certificados de servidor, chame o método `listServerCertificates` do AmazonIdentityManagementClient com um ListServerCertificatesRequest. Ele retorna um ListServerCertificatesResult.

Chame o método `getServerCertificateMetadataList` do objeto `ListServerCertificateResult` para obter uma lista de objetos ServerCertificateMetadata usados por você para obter informações sobre cada certificado.

Os resultados podem ser truncados. Se o método `ListServerCertificateResult` do objeto `getIsTruncated` retornar `true`, chame o método `ListServerCertificatesRequest` do objeto `setMarker` e o use para chamar `listServerCertificates` novamente a fim de obter o próximo lote de resultados.

Importações

```
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;
```

Código da

```
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;
    }
```

```
}
```

Veja o exemplo completo no GitHub.

Atualizar um certificado de servidor

Você pode atualizar o nome ou o caminho de um certificado de servidor chamando o método `updateServerCertificate` do AmazonIdentityManagementClient. Ele utiliza um objeto UpdateServerCertificateRequest definido com o nome atual do certificado de servidor e um novo nome ou caminho a ser usado.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Excluir um certificado de servidor

Para excluir um certificado de servidor, chame o método `deleteServerCertificate` do AmazonIdentityManagementClient com um DeleteServerCertificateRequest contendo o nome do certificado.

Importações

```
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;
```

Código da

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

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

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

Veja o exemplo completo no GitHub.

Mais informações

- Trabalhar com certificados de servidor no Guia do Usuário do IAM
- GetServerCertificate na Referência de API do IAM
- ListServerCertificates na Referência de API do IAM
- UpdateServerCertificate na Referência de API do IAM
- DeleteServerCertificate na Referência de API do IAM
- Guia do usuário do ACM

Lambda Exemplos de usando a AWS SDK for Java

Esta seção apresenta exemplos de como programar o Lambda usando o AWS SDK for Java.

> **Note**
>
> Entre os exemplos está somente o código necessário para demonstrar cada técnica. O código de exemplo completo está disponível no GitHub. A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

Invocar, listar e excluir funções do Lambda

Esta seção fornece exemplos de programação com o cliente de serviço do Lambda usando o AWS SDK for Java. Para saber como criar uma função do Lambda, consulte Como criar funções do AWS Lambda.

Tópicos

Invocar uma função

É possível invocar uma função do Lambda criando um objeto AWSLambda e invocando seu método `invoke`. Crie um objeto InvokeRequest para especificar informações adicionais, como o nome da função e a carga útil a serem transmitidas para a função do Lambda. Os nomes de função aparecem como arn:aws:lambda:us-east-1:555556330391:function:HelloFunction. É possível recuperar o valor examinando a função no Console de gerenciamento da AWS.

Para transmitir dados de carga para uma função, invoque o método `withPayload` do objeto InvokeRequest e especifique uma String no formato JSON, conforme mostrado no exemplo de código a seguir.

Importações

```
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;
```

Código da

O exemplo de código a seguir demonstra como invocar uma função do 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());
```

Veja o exemplo completo no GitHub.

Listar as funções

Crie um objeto AWSLambda e invoque seu método `listFunctions`. Este método retorna um objeto ListFunctionsResult. É possível invocar o método `getFunctions` desse objeto para retornar uma lista de objetos FunctionConfiguration. É possível percorrer a lista para recuperar informações sobre as funções. Por exemplo, o exemplo de código Java a seguir mostra como obter cada nome de função.

Importações

```
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;
```

Código da

O exemplo de código Java a seguir demonstra como recuperar uma lista de nomes de função do Lambda.

```
        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);
        }
```

Veja o exemplo completo no GitHub.

Excluir uma função

Crie um objeto AWSLambda e invoque seu método `deleteFunction`. Crie um objeto DeleteFunctionRequest e transmita-o ao método `deleteFunction`. Esse objeto contém informações como o nome da função a ser excluída. Os nomes de função aparecem como arn:aws:lambda:us-east-1:555556330391:function:HelloFunction. É possível recuperar o valor examinando a função no Console de gerenciamento da AWS.

Importações

```
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;
```

Código da

O seguinte código Java demonstra como excluir uma função do Lambda.

```
        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);
        }
```

Veja o exemplo completo no GitHub.

Amazon Pinpoint Exemplos de usando a AWS SDK for Java

Esta seção apresenta exemplos de como programar o Amazon Pinpoint usando o AWS SDK for Java.

> **Note**
>
> Entre os exemplos está somente o código necessário para demonstrar cada técnica. O código de exemplo completo está disponível no GitHub. A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

Criar e excluir aplicativos no Amazon Pinpoint

Um aplicativo é um projeto do Amazon Pinpoint no qual você define o público para um aplicativo distinto e envolve esse público com mensagens personalizadas. Os exemplos nesta página demonstram como criar um novo aplicativo ou excluir um existente.

Criar um aplicativo

Crie um aplicativo no Amazon Pinpoint fornecendo um nome de aplicativo ao objeto CreateAppRequest e passando esse objeto para o método `createApp` do AmazonPinpointClient.

Importações

```
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;
```

Código da

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

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

Veja o exemplo completo no GitHub.

Excluir um aplicativo

Para excluir um aplicativo, chame a solicitação `deleteApp` do AmazonPinpointClient com um objeto DeleteAppRequest, que é definido com o nome do aplicativo a ser excluído.

Importações

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

Código da

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

pinpoint.deleteApp(deleteRequest);
```

Veja o exemplo completo no GitHub.

Mais informações

- Apps na Referência de API do Amazon Pinpoint
- App na Referência de API do Amazon Pinpoint

Criar endpoints no Amazon Pinpoint

Um endpoint identifica exclusivamente um dispositivo de usuário ao qual você pode enviar notificações por push com o Amazon Pinpoint. Se o seu aplicativo tem suporte ao Amazon Pinpoint ativado, ele registra automaticamente um endpoint com esse Amazon Pinpoint quando um novo

usuário o abre. O exemplo a seguir demonstra como adicionar um novo endpoint de maneira programática.

Criar um endpoint

Crie um endpoint no Amazon Pinpoint fornecendo os dados do endpoint em um objeto EndpointRequest.

Importações

```
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;
```

Código da

```
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);
```

Em seguida, crie um objeto UpdateEndpointRequest com o objeto EndpointRequest. Por fim, passe o objeto UpdateEndpointRequest para o método `updateEndpoint` do AmazonPinpointClient.

Código da

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

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

Veja o exemplo completo no GitHub.

Mais informações

- Adicionar endpoint no Guia do desenvolvedor do Amazon Pinpoint
- Endpoint na Referência de API do Amazon Pinpoint

Criar segmentos no Amazon Pinpoint

Um segmento de usuário representa um subconjunto de seus usuários com base em características compartilhadas, como a última vez em que os usuários abriram seu aplicativo ou qual dispositivo usam. O exemplo a seguir demonstra como definir um segmento dos usuários.

Criar um segmento

Crie um segmento no Amazon Pinpoint definindo dimensões do segmento em um objeto SegmentDimensions.

Importações

```
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;
```

Código da

```
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);
```

Em seguida, defina o objeto SegmentDimensions em um WriteSegmentRequest usado para criar um objeto CreateSegmentRequest. Depois disso, passe o objeto CreateSegmentRequest para o método `createSegment` do AmazonPinpointClient.

Código da

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

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

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

Veja o exemplo completo no GitHub.

Mais informações

- Segmentos do Amazon Pinpoint no Guia do usuário do Amazon Pinpoint
- Criar segmentos, no Guia do desenvolvedor do Amazon Pinpoint
- Segmentos na Referência de API do Amazon Pinpoint
- Segmento na Referência de API do Amazon Pinpoint

Criar campanhas no Amazon Pinpoint

Você pode usar campanhas para ajudar a aumentar o envolvimento entre seu aplicativo e seus usuários. Você pode criar uma campanha para alcançar um segmento específico dos seus usuários com mensagens personalizadas ou promoções especiais. Este exemplo demonstra como criar uma nova campanha padrão que envia uma notificação push personalizada para um segmento especificado.

Criar uma campanha

Antes de criar uma nova campanha, você deve definir uma programação e uma mensagem e, em seguida, definir esses valores em um objeto WriteCampaignRequest.

Importações

```
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;
```

Código da

```
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);
```

Em seguida, crie uma nova campanha no Amazon Pinpoint fornecendo o WriteCampaignRequest com a configuração da campanha para um objeto CreateCampaignRequest. Por fim, passe o objeto CreateCampaignRequest para o método `createCampaign` do AmazonPinpointClient.

Código da

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

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

Veja o exemplo completo no GitHub.

Mais informações

- Campanhas do Amazon Pinpoint no Guia do usuário do Amazon Pinpoint
- Criar campanhas no Guia do desenvolvedor do Amazon Pinpoint
- Campanhas na Referência de API do Amazon Pinpoint
- Campanha na Referência de API do Amazon Pinpoint
- Atividades de campanha na Referência de API do Amazon Pinpoint
- Versões da campanha na Referência de API do Amazon Pinpoint
- Versão da campanha na Referência de API do Amazon Pinpoint

Atualizar canais no Amazon Pinpoint

Um canal define os tipos de plataformas para as quais você pode entregar mensagens. Este exemplo mostra como usar o canal APNs para enviar uma mensagem.

Atualizar um canal

Ativar um canal no Amazon Pinpoint fornecendo um ID de aplicativo e um objeto de solicitação do tipo de canal que você quer atualizar. Este exemplo atualiza o canal APNs, que requer o objeto

APNSChannelRequest. Defina-os em UpdateApnsChannelRequest e passe o objeto para o método `updateApnsChannel` do AmazonPinpointClient.

Importações

```
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;
```

Código da

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

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

Veja o exemplo completo no GitHub.

Mais informações

- Canais do Amazon Pinpoint no Guia do usuário do Amazon Pinpoint
- Canal ADM na Referência de API do Amazon Pinpoint
- Canal do APNs na Referência de API do Amazon Pinpoint
- Canal Sandbox de APNs na Referência de API do Amazon Pinpoint
- Canal VoIP de APNs na Referência de API do Amazon Pinpoint
- Canal APNs VoIP Sandbox na Referência de API do Amazon Pinpoint
- Canal Baidu na Referência de API do Amazon Pinpoint
- Canal de e-mail na Referência de API do Amazon Pinpoint
- Canal GCM na Referência de API do Amazon Pinpoint

- Canal SMS na Referência de API do Amazon Pinpoint

Amazon S3 Exemplos de usando a AWS SDK for Java

Esta seção apresenta exemplos de como programar o Amazon S3 usando o AWS SDK for Java.

> **Note**
>
> Entre os exemplos está somente o código necessário para demonstrar cada técnica. O código de exemplo completo está disponível no GitHub. A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

- Criar, listar e excluir buckets do Amazon S3
- Realizar operações em objetos do Amazon S3
- Gerenciar permissões de acesso ao Amazon S3 para buckets e objetos
- Gerenciar acesso a buckets do Amazon S3 usando políticas de bucket
- Usar o TransferManager em operações do Amazon S3
- Configurar um bucket do Amazon S3 como um site
- Usar criptografia do lado do cliente do Amazon S3

Criar, listar e excluir buckets do Amazon S3

Cada objeto (arquivo) no Amazon S3 deve residir em um bucket, que representa um conjunto (contêiner) de objetos. Cada bucket é conhecido por uma chave (nome), que deve ser exclusiva. Para obter informações detalhadas sobre os buckets e suas configurações, consulte Trabalhar com buckets do Amazon S3 Guia do usuário do Amazon Simple Storage Service.

> **Note**
>
> Melhor prática
> Recomendamos habilitar a regra de ciclo de vida AbortIncompleteMultipartUpload nos buckets do Amazon S3.

Essa regra leva o Amazon S3 a anular multipart uploads que não sejam concluídos dentro de um número específico de dias depois de serem iniciados. Quando o limite de tempo definido é excedido, o Amazon S3 anula o upload e exclui os dados de uploads incompletos. Para obter mais informações, consulte Configuração do ciclo de vida de um bucket com versionamento no Guia do usuário do Amazon S3.

> **Note**
>
> Esses exemplos de código pressupõem que você entenda o material em Usar o AWS SDK for Java e tenha configurado credenciais da AWS padrão usando as informações em Configurar credenciais e região da AWS para desenvolvimento.

Criar um bucket

Use o método `createBucket` do cliente AmazonS3. O novo bucket é retornado. O método `createBucket` lançará uma exceção se o bucket já existir.

> **Note**
>
> Para verificar se um bucket já existe antes de tentar criar um com o mesmo nome, chame o método `doesBucketExist`. Isso retornará `true` se o bucket existir e, do contrário, `false`.

Importações

```
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;
```

Código da

```
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;
```

Veja o exemplo completo no GitHub.

Listar buckets

Use o método `listBucket` do cliente AmazonS3. Se for bem-sucedido, uma lista de buckets será retornada.

Importações

```
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;
```

Código da

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

Veja o exemplo completo no GitHub.

Excluir um Bucket

Para excluir um bucket do Amazon S3, você deve verificar se o bucket está vazio, ou isso resultará em um erro. Se tiver um bucket versionado, você também deverá excluir todos os objetos versionados associados ao bucket.

> **Note**
>
> O exemplo completo inclui todas essas etapas em ordem, fornecendo uma solução completa para excluir um bucket do Amazon S3 e o conteúdo.

Tópicos

- Remover objetos de um bucket não versionado antes de excluí-lo
- Remover objetos de um bucket versionado antes de excluí-lo
- Excluir um bucket vazio

Remover objetos de um bucket não versionado antes de excluí-lo

Use o método `listObjects` de cliente do AmazonS3 para recuperar a lista de objetos e `deleteObject` para excluir cada um.

Importações

```
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;
```

Código da

```
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;
    }
}
```

Veja o exemplo completo no GitHub.

Remover objetos de um bucket versionado antes de excluí-lo

Se estiver usando um bucket versionado, também será necessário remover todas as versões armazenadas dos objetos no bucket para o bucket ser excluído.

Usando um padrão semelhante ao usado ao remover objetos dentro de um bucket, remova objetos versionados usando o método `listVersions` de cliente do AmazonS3 para listar todos os objetos versionados e `deleteVersion` para excluir cada um.

Importações

```
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;
```

Código da

```
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;
    }
}
```

Veja o exemplo completo no GitHub.

Excluir um bucket vazio

Assim que remover os objetos de um bucket (inclusive todos os objetos versionados), será possível excluir o bucket em si usando o método `deleteBucket` de cliente do AmazonS3.

Importações

```
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;
```

Código da

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

Veja o exemplo completo no GitHub.

Realizar operações em objetos do Amazon S3

Um objeto do Amazon S3 representa um arquivo ou um conjunto de dados. Cada objeto deve residir em um bucket.

> **Note**
>
> Esses exemplos de código pressupõem que você entenda o material em Usar o AWS SDK for Java e tenha configurado credenciais da AWS padrão usando as informações em Configurar credenciais e região da AWS para desenvolvimento.

Tópicos

Fazer upload de um objeto

Use o método `putObject` de cliente do AmazonS3 fornecendo um nome de bucket, um nome de chave e um arquivo para upload. O bucket deve existir ou isso resultará em um erro.

Importações

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

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Listar objetos

Para obter uma lista de objetos em um bucket, use o método `listObjects` de cliente do AmazonS3 fornecendo o nome de um bucket.

O método `listObjects` retorna um objeto ObjectListing que fornece informações sobre os objetos no bucket. Para listar os nomes de objeto (chaves), use o método `getObjectSummaries` para obter uma lista de objetos S3ObjectSummary, cada um dos quais representa um único objeto no bucket. Depois disso, chame o método `getKey` para recuperar o nome do objeto.

Importações

```
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;
```

Código da

```
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());
}
```

Veja o exemplo completo no GitHub.

Fazer download de um objeto

Use o método `getObject` de cliente do AmazonS3 passando o nome de um bucket e o objeto para fazer download. Se bem-sucedido, o método retornará um S3Object. O bucket especificado e a chave de objeto devem existir ou isso resultará em um erro.

É possível obter o conteúdo do objeto chamando `getObjectContent` no `S3Object`. Isso retorna um S3ObjectInputStream que se comporta como um objeto `InputStream` do Java padrão.

O exemplo a seguir faz download de um objeto do S3 e salva o conteúdo em um arquivo (usando o mesmo nome da chave do objeto).

Importações

```
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;
```

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Copiar, mover ou renomear objetos

É possível copiar um objeto de um bucket para outro usando o método `copyObject` de cliente do AmazonS3. Ele utiliza o nome do bucket do qual será feita a cópia, o objeto a ser copiado e o nome do bucket de destino.

Importações

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Código da

```
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!");
```

Veja o exemplo completo no GitHub.

> **Note**
>
> Você pode usar `copyObject` com deleteObject para migrar ou renomear um objeto copiando primeiro o objeto para um novo nome (é possível usar o mesmo bucket na origem e no destino) e excluindo o objeto do local anterior.

Excluir um objeto

Use o método `deleteObject` de cliente do AmazonS3, passando o nome de um bucket e o objeto a ser excluído. O bucket especificado e a chave de objeto devem existir ou isso resultará em um erro.

Importações

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Excluir vários objetos de uma só vez

Usando o método `deleteObjects` de cliente do AmazonS3, você pode excluir vários objetos do mesmo bucket passando seus nomes para o método link:sdk-for-java/v1/reference/com/amazonaws/services/s3/model/DeleteObjectsRequest.html.

Importações

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

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Gerenciar permissões de acesso ao Amazon S3 para buckets e objetos

É possível usar listas de controle de acesso (ACLs) para objetos e buckets do Amazon S3 para controle refinado sobre os recursos do Amazon S3.

> **Note**
>
> Esses exemplos de código pressupõem que você entenda o material em Usar o AWS SDK for Java e tenha configurado credenciais da AWS padrão usando as informações em Configurar credenciais e região da AWS para desenvolvimento.

Obter a lista de controle de acesso para um bucket

Para obter a ACL atual de um bucket, chame o método `getBucketAcl` do AmazonS3 passando o nome do bucket para consulta. Esse método retorna um objeto AccessControlList. Para obter cada concessão de acesso na lista, chame o método `getGrantsAsList`, que retornará uma lista de objetos Grant do Java padrão.

Importações

```
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;
```

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Definir a lista de controle de acesso para um bucket

Para adicionar ou modificar permissões para uma ACL de um bucket, chame o método `setBucketAcl` do AmazonS3. Ele utiliza um objeto AccessControlList que contém uma lista de favorecidos e níveis de acesso a serem definidos.

Importações

```
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;
```

Código da

```
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**
>
> É possível fornecer o identificador exclusivo do favorecido diretamente usando a classe Grantee ou usar a classe EmailAddressGrantee para definir o favorecido por e-mail, como fizemos aqui.

Veja o exemplo completo no GitHub.

Obter a lista de controle de acesso para um objeto

Para obter a ACL atual de um objeto, chame o método `getObjectAcl` do AmazonS3, passando o nome do bucket e o nome do objeto para consulta. Assim como `getBucketAcl`, esse método retorna um objeto AccessControlList que você pode usar para examinar cada Grant.

Importações

```
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;
```

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Definir a lista de controle de acesso para um objeto

Para adicionar ou modificar permissões para uma ACL de um objeto, chame o método `setObjectAcl` do AmazonS3. Ele utiliza um objeto AccessControlList que contém uma lista de favorecidos e níveis de acesso a serem definidos.

Importações

```
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;
```

Código da

```
    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**
>
> É possível fornecer o identificador exclusivo do favorecido diretamente usando a classe Grantee ou usar a classe EmailAddressGrantee para definir o favorecido por e-mail, como fizemos aqui.

Veja o exemplo completo no GitHub.

Mais informações

- GET Bucket acl na Referência de API do Amazon S3
- PUT Bucket acl na Referência de API do Amazon S3
- GET Object acl na Referência de API do Amazon S3
- PUT Object acl na Referência de API do Amazon S3

Gerenciar acesso a buckets do Amazon S3 usando políticas de bucket

Você pode definir, obter ou excluir uma política de bucket para gerenciar o acesso aos buckets do Amazon S3.

Definir uma política de bucket

Você pode definir a política de bucket para um determinado bucket do S3 ao:

- Chamar o `setBucketPolicy` de cliente do AmazonS3 e fornecer um SetBucketPolicyRequest
- Definir a política diretamente usando a sobrecarga `setBucketPolicy` que utiliza um nome de bucket e o texto da política (em formato JSON)

Importações

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
```

Código da

```
    s3.setBucketPolicy(bucket_name, policy_text);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Usar a classe Policy para gerar ou validar uma política

Ao fornecer uma política de bucket para `setBucketPolicy`, você pode fazer o seguinte:

- Especificar a política diretamente como uma string de texto formatado em JSON
- Compilar a política usando a classe Policy

Usando a classe `Policy`, não é necessário se preocupar com a formatação correta da string de texto. Para obter o texto da política JSON da classe `Policy`, use o método `toJson`.

Importações

```
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;
```

Código da

```
        new Statement(Statement.Effect.Allow)
                .withPrincipals(Principal.AllUsers)
                .withActions(S3Actions.GetObject)
                .withResources(new Resource(
                        "{region-arn}s3:::" + bucket_name + "/*")));
return bucket_policy.toJson();
```

A classe `Policy` também oferece um método `fromJson` que pode tentar compilar uma política usando uma string JSON passada. O método a valida para garantir que o texto possa ser transformado em uma estrutura de política válida e falhará com um `IllegalArgumentException` se o texto da política for inválido.

```
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());
}
```

Você pode usar essa técnica para pré-validar uma política lida de um arquivo ou outros meios.

Veja o exemplo completo no GitHub.

Obter uma política de bucket

Para recuperar a política de um bucket do Amazon S3, chame o método `getBucketPolicy` de cliente do AmazonS3, passando o nome do bucket do qual obter a política.

Importações

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

Código da

```
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 o bucket nomeado não existir, se você não tiver acesso a ele, ou se ele não tiver uma política de bucket, um `AmazonServiceException` será lançado.

Veja o exemplo completo no GitHub.

Excluir uma política de bucket

Para excluir uma política de bucket, chame o `deleteBucketPolicy` de cliente do AmazonS3, fornecendo o nome do bucket.

Importações

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

Código da

```
try {
    s3.deleteBucketPolicy(bucket_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Esse método será bem-sucedido, mesmo se o bucket ainda não tiver uma política. Se você especificar um nome de bucket não existente ou se não tiver acesso ao bucket, um `AmazonServiceException` será lançado.

Veja o exemplo completo no GitHub.

Mais informações

- Visão geral da linguagem de políticas de acesso no Guia do usuário do Amazon Simple Storage Service
- Exemplos de políticas de bucket no Guia do usuário do Amazon Simple Storage Service

Usar o TransferManager em operações do Amazon S3

Você pode usar a classe TransferManager do AWS SDK for Java para transferir arquivos de maneira confiável do ambiente local para Amazon S3 e copiar objetos de um local do S3 para outro. O `TransferManager` pode saber o andamento de uma transferência e pausar ou retomar uploads e downloads.

> **Note**
>
> Melhor prática
> Recomendamos habilitar a regra de ciclo de vida AbortIncompleteMultipartUpload nos buckets do Amazon S3.
> Essa regra leva o Amazon S3 a anular multipart uploads que não sejam concluídos dentro de um número específico de dias depois de serem iniciados. Quando o limite de tempo definido é excedido, o Amazon S3 anula o upload e exclui os dados de uploads incompletos.
> Para obter mais informações, consulte Configuração do ciclo de vida de um bucket com versionamento no Guia do usuário do Amazon S3.

> **Note**
>
> Esses exemplos de código pressupõem que você entenda o material em Usar o AWS SDK for Java e tenha configurado credenciais da AWS padrão usando as informações em Configurar credenciais e região da AWS para desenvolvimento.

Fazer upload de arquivos e diretórios

O TransferManager pode fazer upload de arquivos, listas de arquivos e diretórios para todos os buckets do Amazon S3 criados anteriormente.

Tópicos

Fazer upload de um único arquivo

Chame o método `upload` do TransferManager, fornecendo um nome de bucket do Amazon S3, um nome de chave (objeto) e um objeto File do Java padrão que represente o arquivo para upload.

Importações

```
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;
```

Código da

```
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();
```

O método `upload` retorna imediatamente, fornecendo um objeto `Upload` a ser usado para verificar o estado de transferência ou aguardar a conclusão.

Consulte Aguardar a conclusão de uma transferência para obter informações sobre como usar `waitForCompletion` para concluir com êxito uma transferência antes de chamar o método `shutdownNow` do TransferManager. Enquanto aguarda a conclusão da transferência, você pode sondar ou escutar atualizações sobre o status e o progresso. Consulte Obter status de transferência e progresso para obter mais informações.

Veja o exemplo completo no GitHub.

Fazer upload de uma lista de arquivos

Para fazer upload de vários arquivos em uma única operação, chame o método `uploadFileList` do TransferManager, fornecendo o seguinte:

- Um nome do bucket do Amazon S3
- Um prefixo de chaves a ser acrescentado aos nomes dos objetos criados (o caminho dentro do bucket no qual colocar os objetos)
- Um objeto File que representa o diretório relativo do qual criar caminhos de arquivo
- Um objeto List contendo um conjunto de objetos File para upload

Importações

```
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;
```

Código da

```
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();
```

Consulte Aguardar a conclusão de uma transferência para obter informações sobre como usar `waitForCompletion` para concluir com êxito uma transferência antes de chamar o método `shutdownNow` do TransferManager. Enquanto aguarda a conclusão da transferência, você pode sondar ou escutar atualizações sobre o status e o progresso. Consulte Obter status de transferência e progresso para obter mais informações.

O objeto MultipleFileUpload retornado por `uploadFileList` pode ser usado para consultar o estado da transferência ou o progresso. Consulte Sondar o progresso atual de uma transferência e Obter o progresso da transferência com um ProgressListener para obter mais informações.

Você também pode usar o método `MultipleFileUpload` de `getSubTransfers` para obter os objetos `Upload` individuais de cada arquivo transferido. Para obter mais informações, consulte Obter o progresso de subtransferências.

Veja o exemplo completo no GitHub.

Fazer upload de um diretório

É possível usar o método `uploadDirectory` do TransferManager para fazer upload de um diretório de arquivos inteiro, com a opção de copiar arquivos em subdiretórios de maneira recursiva. Você fornece um nome de bucket do Amazon S3, um prefixo de chaves do S3, um objeto File representando o diretório local para cópia e um valor `boolean` que indica se deseja copiar subdiretórios de maneira recursiva (verdadeiro ou falso).

Importações

```
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;
```

Código da

```
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();
```

Consulte Aguardar a conclusão de uma transferência para obter informações sobre como usar `waitForCompletion` para concluir com êxito uma transferência antes de chamar o método `shutdownNow` do TransferManager. Enquanto aguarda a conclusão da transferência, você pode sondar ou escutar atualizações sobre o status e o progresso. Consulte Obter status de transferência e progresso para obter mais informações.

O objeto MultipleFileUpload retornado por `uploadFileList` pode ser usado para consultar o estado da transferência ou o progresso. Consulte Sondar o progresso atual de uma transferência e Obter o progresso da transferência com um ProgressListener para obter mais informações.

Você também pode usar o método `MultipleFileUpload` de `getSubTransfers` para obter os objetos `Upload` individuais de cada arquivo transferido. Para obter mais informações, consulte Obter o progresso de subtransferências.

Veja o exemplo completo no GitHub.

Fazer download de arquivos ou diretórios

Use a classe TransferManager para fazer download de um único arquivo (objeto do Amazon S3) ou de um diretório (um nome de bucket do Amazon S3 seguido de um prefixo de objeto) do Amazon S3.

Tópicos

Fazer download de um único arquivo

Use o método `download` do TransferManager, fornecendo o nome de bucket do Amazon S3 que contém o objeto cujo download você deseja fazer, o nome da chave (objeto) e um objeto File que representa o arquivo a ser criado no sistema local.

Importações

```
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;
```

Código da

```
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();
```

Consulte Aguardar a conclusão de uma transferência para obter informações sobre como usar `waitForCompletion` para concluir com êxito uma transferência antes de chamar o método `shutdownNow` do TransferManager. Enquanto aguarda a conclusão da transferência, você pode

sondar ou escutar atualizações sobre o status e o progresso. Consulte Obter status de transferência e progresso para obter mais informações.

Veja o exemplo completo no GitHub.

Fazer download de um diretório

Para fazer download de um conjunto de arquivos que compartilham um mesmo prefixo de chaves (semelhante a um diretório em um sistema de arquivos) do Amazon S3, use o método `downloadDirectory` do TransferManager. O método utiliza o nome de bucket do Amazon S3 que contém os objetos cujo download você deseja fazer, o prefixo do objeto compartilhado por todos os objetos e um objeto File que representa o diretório para fazer download dos arquivos no sistema local. Se ainda não existir, o diretório nomeado será criado.

Importações

```
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;
```

Código da

```
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();
```

Consulte Aguardar a conclusão de uma transferência para obter informações sobre como usar `waitForCompletion` para concluir com êxito uma transferência antes de chamar o método `shutdownNow` do TransferManager. Enquanto aguarda a conclusão da transferência, você pode sondar ou escutar atualizações sobre o status e o progresso. Consulte Obter status de transferência e progresso para obter mais informações.

Veja o exemplo completo no GitHub.

Copiar objetos

Para copiar um objeto de um bucket do S3 para outro, use o método `copy` do TransferManager.

Importações

```
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;
```

Código da

```
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();
```

Veja o exemplo completo no GitHub.

Aguardar a conclusão de uma transferência

Se o aplicativo (ou thread) puder bloquear até a conclusão da transferência, você poderá usar o método `waitForCompletion` da interface Transfer para bloquear até a transferência estar concluída ou ocorrer uma exceção.

```
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);
}
```

Você obterá o progresso de transferências se sondar eventos antes de chamar `waitForCompletion`, implementar um mecanismo de sondagem em um thread separado ou receber atualizações de progresso de maneira assíncrona usando um ProgressListener.

Veja o exemplo completo no GitHub.

Obter status da transferência e progresso

Cada uma das classes retornadas pelos métodos `upload*`, `download*` e `copy` do TransferManager retorna uma instância de uma das classes a seguir, dependendo da operação ser de arquivo único ou de vários arquivos.

Classe	Retornado por
Copiar	`copy`
Baixar	`download`
MultipleFileDownload	`downloadDirectory`
Carregar	`upload`

Classe	Retornado por
MultipleFileUpload	`uploadFileList` , `uploadDirectory`

Todas essas classes implementam a interface Transfer. O `Transfer` oferece métodos úteis para obter o progresso de uma transferência, pausar ou retomar a transferência, além de obter o status atual ou final da transferência.

Tópicos

- Sondar o progresso atual de uma transferência
- Obter o progresso da transferência com um ProgressListener
- Obter o progresso de subtransferências

Sondar o progresso atual de uma transferência

Este loop imprime o progresso de uma transferência, examina o progresso atual durante a execução e, quando concluído, imprime o estado final.

Importações

```
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;
```

Código da

```
// 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);
```

Veja o exemplo completo no GitHub.

Obter o progresso da transferência com um ProgressListener

Você pode anexar um ProgressListener a qualquer transferência usando o método `addProgressListener` da interface Transfer.

Um ProgressListener exige somente um método, `progressChanged`, que utiliza um objeto ProgressEvent. Você pode usar o objeto para obter o total de bytes da operação chamando o método `getBytes` e o número de bytes transferidos até o momento chamando `getBytesTransferred`.

Importações

```
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;
```

Código da

```
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();
```

Veja o exemplo completo no GitHub.

Obter o progresso de subtransferências

A classe MultipleFileUpload pode retornar informações sobre as subtransferências chamando o método `getSubTransfers`. Isso retorna um Conjunto de objetos Upload que fornecem o status de transferência individual e o progresso de cada subtransferência.

Importações

```
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;
```

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Mais informações

- Chaves de objeto no Guia do usuário do Amazon Simple Storage Service

Configurar um bucket do Amazon S3 como um site

É possível configurar um bucket do Amazon S3 para se comportar como um site. Para isso, você precisa definir a configuração do site.

> **Note**
>
> Esses exemplos de código pressupõem que você entenda o material em Usar o AWS SDK for Java e tenha configurado credenciais da AWS padrão usando as informações em Configurar credenciais e região da AWS para desenvolvimento.

Definir uma configuração do site de um bucket

Para definir a configuração do site de um bucket do Amazon S3, chame o método `setWebsiteConfiguration` do AmazonS3 com o nome do bucket para definir a configuração e um objeto BucketWebsiteConfiguration contendo a configuração do site do bucket.

Configurar um documento de índice é obrigatório; todos os outros parâmetros são opcionais.

Importações

```
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;
```

Código da

```
    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**
>
> Definir a configuração de um site não modifica as permissões de acesso do bucket. Para tornar os arquivos visíveis na web, você também precisa definir uma política de bucket que permite acesso de leitura público aos arquivos no bucket. Para obter mais informações, consulte Gerenciar acesso aos buckets do Amazon S3 usando políticas de bucket.

Veja o exemplo completo no GitHub.

Obter uma configuração do site de um bucket

Para obter a configuração do site de um bucket do Amazon S3, chame o método `getWebsiteConfiguration` do AmazonS3 com o nome do bucket cuja configuração recuperar.

A configuração será retornada como um objeto BucketWebsiteConfiguration. Se não houver configuração de site para o bucket, `null` será retornado.

Importações

```
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;
```

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Excluir uma configuração do site de um bucket

Para excluir a configuração do site de um bucket do Amazon S3, chame o método `deleteWebsiteConfiguration` do AmazonS3 com o nome do bucket cuja configuração será excluída.

Importações

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

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Mais informações

- PUT Bucket website na Referência de API do Amazon S3
- GET Bucket website na Referência de API do Amazon S3
- DELETE Bucket website na Referência de API do Amazon S3

Usar criptografia do lado do cliente do Amazon S3

Criptografar dados usando o cliente de criptografia do Amazon S3 é uma maneira de fornecer uma camada adicional de proteção para informações confidenciais armazenadas no Amazon S3. Os exemplos nesta seção demonstram como criar e configurar o cliente de criptografia do Amazon S3 para seu aplicativo.

Se você for novo em criptografia, consulte Conceitos básicos de criptografia no Guia do desenvolvedor do AWS KMS para uma visão geral básica dos termos e algoritmos de criptografia. Para obter informações sobre suporte à criptografia em todos os SDKs da AWS, consulte Suporte do AWS SDK à criptografia do lado do cliente do Amazon S3 na Referência geral da Amazon Web Services.

> **Note**
>
> Esses exemplos de código pressupõem que você entenda o material em Usar o AWS SDK for Java e tenha configurado credenciais da AWS padrão usando as informações em Configurar credenciais e região da AWS para desenvolvimento.

Se você estiver usando a versão 1.11.836 ou anterior do AWS SDK for Java, consulte Migração do cliente de criptografia do Amazon S3 para obter informações sobre como migrar seus aplicativos para versões posteriores. Se você não conseguir migrar, consulte este exemplo completo no GitHub.

Caso contrário, se você estiver usando a versão 1.11.837 ou posterior do AWS SDK for Java, explore os tópicos de exemplo listados abaixo para usar a criptografia do lado do cliente do Amazon S3.

Tópicos

Criptografia do lado do cliente do Amazon S3 com chaves mestras do lado do cliente

Os exemplos a seguir usam a classe AmazonS3EncryptionClientV2Builder para criar um cliente do Amazon S3 com criptografia do lado do cliente ativada. Uma vez ativado, todo objeto que você enviar para o Amazon S3 usando este cliente será criptografado. Todos os objetos obtidos no Amazon S3 por meio deste cliente são automaticamente descriptografados.

> **Note**
>
> Os exemplos a seguir demonstram como usar a criptografia do lado do cliente do Amazon S3 com as chaves mestres clientes gerenciadas do cliente. Para aprender a usar criptografia com chaves gerenciadas pelo AWS KMS, consulte Criptografia do lado do cliente do Amazon S3 com chaves gerenciadas do AWS KMS.

Você pode escolher um dos dois modos de criptografia ao ativar a criptografia do lado do cliente do Amazon S3: rigorosa autenticada ou autenticada. As seções a seguir mostram como ativar cada tipo. Para saber quais algoritmos cada modo usa, consulte a definição CryptoMode.

Requer importações

Importe as seguintes classes para esses exemplos.

Importações

```
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;
```

Criptografia rigorosa autenticada

A criptografia rigorosa autenticada é o modo padrão se nenhum `CryptoMode` for especificado.

Para ativar explicitamente este modo, especifique o valor `StrictAuthenticatedEncryption` no método `withCryptoConfiguration`.

> **Note**
>
> Para usar criptografia autenticada do lado do cliente, você precisa incluir o arquivo Bouncy Castle jar mais recente no caminho de classe do seu aplicativo.

Código da

```
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");
```

Modo de criptografia autenticada

Quando você usa o modo `AuthenticatedEncryption`, um algoritmo de encapsulamento de chave aprimorado é aplicado durante a criptografia. Ao descriptografar nesse modo, o algoritmo verifica a integridade do objeto descriptografado e lança uma exceção se a verificação falhar. Para obter mais detalhes sobre como a criptografia autenticada funciona, consulte a postagem do blog Criptografia autenticada no lado do cliente do Amazon S3.

> **Note**
>
> Para usar criptografia autenticada do lado do cliente, você precisa incluir o arquivo Bouncy Castle jar mais recente no caminho de classe do seu aplicativo.

Para ativar este modo, especifique o valor `AuthenticatedEncryption` no método `withCryptoConfiguration`.

Código da

```
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");
```

Criptografia do lado do cliente do Amazon S3 com chaves gerenciadas do AWS KMS

Os exemplos a seguir usam a classe AmazonS3EncryptionClientV2Builder para criar um cliente do Amazon S3 com criptografia do lado do cliente ativada. Uma vez configurado, todo objeto que você enviar para o Amazon S3 usando este cliente será criptografado. Todos os objetos obtidos no Amazon S3 por meio deste cliente são automaticamente descriptografados.

> **Note**
>
> Os exemplos a seguir demonstram como usar a criptografia do lado do cliente do Amazon S3 com as chaves gerenciadas do AWS KMS. Para aprender a usar criptografia com suas próprias chaves, consulte Criptografia do lado do cliente do Amazon S3 com chaves mestras de cliente.

Você pode escolher um dos dois modos de criptografia ao ativar a criptografia do lado do cliente do Amazon S3: rigorosa autenticada ou autenticada. As seções a seguir mostram como ativar cada tipo. Para saber quais algoritmos cada modo usa, consulte a definição CryptoMode.

Requer importações

Importe as seguintes classes para esses exemplos.

Importações

```
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;
```

Criptografia rigorosa autenticada

A criptografia rigorosa autenticada é o modo padrão se nenhum `CryptoMode` for especificado.

Para ativar explicitamente este modo, especifique o valor `StrictAuthenticatedEncryption` no método `withCryptoConfiguration`.

> **Note**
>
> Para usar criptografia autenticada do lado do cliente, você precisa incluir o arquivo Bouncy Castle jar mais recente no caminho de classe do seu aplicativo.

Código da

```
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));
```

Chame o método `putObject` no cliente de criptografia do Amazon S3 para os objetos de upload.

Código da

```
s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
 with a key created in the {console}");
```

Você pode recuperar o objeto usando o mesmo cliente. Este exemplo chama o método `getObjectAsString` para recuperar a string que foi armazenada.

Código da

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

Modo de criptografia autenticada

Quando você usa o modo `AuthenticatedEncryption`, um algoritmo de encapsulamento de chave aprimorado é aplicado durante a criptografia. Ao descriptografar nesse modo, o algoritmo verifica a integridade do objeto descriptografado e lança uma exceção se a verificação falhar. Para obter mais detalhes sobre como a criptografia autenticada funciona, consulte a postagem do blog Criptografia autenticada no lado do cliente do Amazon S3.

> **Note**
>
> Para usar criptografia autenticada do lado do cliente, você precisa incluir o arquivo Bouncy Castle jar mais recente no caminho de classe do seu aplicativo.

Para ativar este modo, especifique o valor `AuthenticatedEncryption` no método `withCryptoConfiguration`.

Código da

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

Configurar o cliente do AWS KMS

O cliente de criptografia do Amazon S3 cria um cliente do AWS KMS por padrão, a menos que um seja explicitamente especificado.

Para definir a região desse cliente do AWS KMS criado automaticamente, defina a`awsKmsRegion`.

Código da

```
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();
```

Se preferir, pode usar seu próprio cliente do AWS KMS para inicializar o cliente de criptografia.

Código da

```
AWSKMS kmsClient = AWSKMSClientBuilder.standard()
        .withRegion(Regions.US_WEST_2);
        .build();

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

Amazon SQS Exemplos de usando a AWS SDK for Java

Esta seção apresenta exemplos de como programar o Amazon SQS usando o AWS SDK for Java.

> **Note**
>
> Entre os exemplos está somente o código necessário para demonstrar cada técnica. O código de exemplo completo está disponível no GitHub. A partir daí, você pode fazer

download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

Trabalhar com filas de mensagens do Amazon SQS

Uma fila de mensagens é o contêiner lógico usado para enviar mensagens de maneira confiável no Amazon SQS. Existem dois tipos de filas: padrão e First-In, First-Out (FIFO – Primeiro a entrar, primeiro a sair). Para saber mais sobre as filas e as diferenças entre esses tipos, consulte o Guia do desenvolvedor do Amazon SQS.

Este tópico descreve como criar, listar, excluir e obter o URL de uma fila do Amazon SQS usando o AWS SDK for Java.

Criar uma fila

Use o método `createQueue` do cliente do AmazonSQS fornecendo um objeto CreateQueueRequest que descreve os parâmetros de fila.

Importações

```
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;
```

Código da

```
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;
    }
}
```

Você pode usar a forma simplificada de `createQueue`, que precisa somente do nome de uma fila, para criar uma fila padrão.

```
sqs.createQueue("MyQueue" + new Date().getTime());
```

Veja o exemplo completo no GitHub.

Listar filas

Para listar as filas do Amazon SQS da conta, chame o método `listQueues` do cliente do AmazonSQS.

Importações

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

Código da

```
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);
}
```

Usar a sobrecarga `listQueues` sem parâmetros retorna todas as filas. Você pode filtrar os resultados retornados passando um objeto `ListQueuesRequest`.

Importações

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

Código da

```
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);
}
```

Veja o exemplo completo no GitHub.

Obter o URL de uma fila

Chame o método `getQueueUrl` do cliente do AmazonSQS.

Importações

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

Código da

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
String queue_url = sqs.getQueueUrl(QUEUE_NAME).getQueueUrl();
```

Veja o exemplo completo no GitHub.

Excluir uma fila

Forneça o URL da fila para o método `deleteQueue` do cliente do AmazonSQS.

Importações

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

Código da

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

Veja o exemplo completo no GitHub.

Mais informações

- Como as filas do Amazon SQS funcionam no Guia do Desenvolvedor do Amazon SQS
- CreateQueue na Referência de API do Amazon SQS
- GetQueueUrl na Referência de API do Amazon SQS
- ListQueues na Referência de API do Amazon SQS
- DeleteQueues na Referência de API do Amazon SQS

Enviar, receber e excluir mensagens do Amazon SQS

Este tópico descreve como enviar, receber e excluir mensagens do Amazon SQS. As mensagens são sempre entregues usando-se uma fila do SQS.

Enviar uma mensagem

Adicione uma mensagem única a uma fila do Amazon SQS chamando o método `sendMessage` do cliente do AmazonSQS. Forneça um objeto SendMessageRequest que contenha o URL da fila, o corpo da mensagem e o valor de atraso opcional (em segundos).

Importações

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

Código da

```
SendMessageRequest send_msg_request = new SendMessageRequest()
        .withQueueUrl(queueUrl)
        .withMessageBody("hello world")
        .withDelaySeconds(5);
sqs.sendMessage(send_msg_request);
```

Veja o exemplo completo no GitHub.

Enviar várias mensagens de uma só vez

Você pode enviar mais de uma mensagem em uma única solicitação. Para enviar várias mensagens, use o método `sendMessageBatch` do cliente do AmazonSQS, que utiliza um SendMessageBatchRequest que contém o URL da fila e uma lista de mensagens (cada uma sendo um SendMessageBatchRequestEntry) a serem enviadas. Você também pode definir um valor de atraso opcional por mensagem.

Importações

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
import com.amazonaws.services.sqs.model.SendMessageBatchRequestEntry;
```

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Receber mensagens

Recupere todas as mensagens que estejam atualmente na fila chamando o método `receiveMessage` do cliente do AmazonSQS, passando o URL da fila. As mensagens são retornadas como uma lista de objetos Message.

Importações

```
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
```

Código da

```
List<Message> messages = sqs.receiveMessage(queueUrl).getMessages();
```

Excluir mensagens depois do recebimento

Após receber uma mensagem e processar o conteúdo, exclua a mensagem da fila enviando o identificador de recebimento da mensagem e o URL da fila para o método `deleteMessage` de cliente do AmazonSQS.

Código da

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

Veja o exemplo completo no GitHub.

Mais informações

- Como as filas do Amazon SQS funcionam no Guia do Desenvolvedor do Amazon SQS
- SendMessage na Referência de API do Amazon SQS
- SendMessageBatch na Referência de API do Amazon SQS
- ReceiveMessage na Referência de API do Amazon SQS
- DeleteMessage na Referência de API do Amazon SQS

Habilitar sondagem longa para filas de mensagens do Amazon SQS

Por padrão, o Amazon SQS usa a sondagem curta, consultando apenas um subconjunto dos servidores (com base na distribuição aleatória ponderada) para determinar se as mensagens estão disponíveis para inclusão na resposta.

A sondagem longa ajuda a reduzir o custo de usar o Amazon SQS reduzindo o número de respostas vazias quando não há mensagens disponíveis para serem retornadas em resposta a uma solicitação ReceiveMessage enviada para uma fila do Amazon SQS e eliminando respostas vazias falsas.

> **Note**
>
> Você pode definir uma frequência de sondagem longa de 1 a 20 segundos.

Habilitar sondagem longa ao criar uma fila

Para habilitar a sondagem longa ao criar uma fila do Amazon SQS, defina o atributo `ReceiveMessageWaitTimeSeconds` no objeto CreateQueueRequest antes de chamar o método `createQueue` de classe do AmazonSQS.

Importações

```
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;
```

Código da

```
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;
    }
}
```

Veja o exemplo completo no GitHub.

Habilitar sondagem longa em uma fila existente

Além de habilitar a sondagem longa ao criar uma fila, você também pode habilitá-la em uma fila existente definindo `ReceiveMessageWaitTimeSeconds` no SetQueueAttributesRequest antes de chamar o método `setQueueAttributes` de classe do AmazonSQS.

Importações

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

Código da

```
SetQueueAttributesRequest set_attrs_request = new SetQueueAttributesRequest()
        .withQueueUrl(queue_url)
        .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");
sqs.setQueueAttributes(set_attrs_request);
```

Veja o exemplo completo no GitHub.

Habilitar sondagem longa no recebimento da mensagem

Você pode habilitar a sondagem longa ao receber uma mensagem definindo o tempo de espera em segundos no ReceiveMessageRequest fornecido por você ao método `receiveMessage` de classe do AmazonSQS.

> **Note**
>
> É necessário verificar se o tempo limite da solicitação do cliente da AWS é maior que o tempo de sondagem longa máximo (20 segundos), de forma que as solicitações `receiveMessage` não expirem enquanto esperam o próximo evento de sondagem.

Importações

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

Código da

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

Veja o exemplo completo no GitHub.

Mais informações

- Sondagem longa do Amazon SQS no Guia do desenvolvedor do Amazon SQS
- CreateQueue na Referência de API do Amazon SQS

- ReceiveMessage na Referência de API do Amazon SQS
- SetQueueAttributes na Referência de API do Amazon SQS

Definir tempo limite de visibilidade no Amazon SQS

Quando for recebida no Amazon SQS, uma mensagem permanecerá na fila até ser excluída para garantir o recebimento. Uma mensagem que foi recebida, mas não excluída, estará disponível em requisições subsequentes depois de um determinado tempo limite de visibilidade para ajudar a evitar que a mensagem seja recebida mais de uma vez antes de ser processada e excluída.

> **Note**
>
> Ao usar filas padrão, o tempo limite de visibilidade não é uma garantia em relação ao recebimento de uma mensagem duas vezes. Se você estiver usando uma fila padrão, verifique se o código pode processar o caso em que a mesma mensagem foi entregue mais de uma vez.

Definir o tempo limite de visibilidade da mensagem para uma única mensagem

Quando tiver recebido uma mensagem, você poderá modificar o tempo limite de visibilidade passando o identificador de recebimento em um ChangeMessageVisibilityRequest passado para o método `changeMessageVisibility` da classe do AmazonSQS.

Importações

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

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Definir o tempo limite de visibilidade da mensagem para várias mensagens de uma só vez

Para definir o tempo limite de visibilidade da mensagem para várias mensagens de uma só vez, crie uma lista de objetos ChangeMessageVisibilityBatchRequestEntry, cada um contendo uma string de ID exclusiva e um identificador de recebimento. Em seguida, passe a lista para o método Amazon SQS da classe de cliente do `changeMessageVisibilityBatch`.

Importações

```
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;
```

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Mais informações

- Tempo limite de visibilidade no Guia do desenvolvedor do Amazon SQS
- SetQueueAttributes na Referência de API do Amazon SQS
- GetQueueAttributes na Referência de API do Amazon SQS
- ReceiveMessage na Referência de API do Amazon SQS
- ChangeMessageVisibility na Referência de API do Amazon SQS
- ChangeMessageVisibilityBatch na Referência de API do Amazon SQS

Usar fila de mensagens mortas no Amazon SQS

O Amazon SQS fornece suporte para filas de mensagens mortas. Uma fila de mensagens mortas é uma fila para a qual outras filas (de origem) podem enviar as mensagens que não são processadas com êxito. Você pode separar e isolar essas mensagens na dead letter queue para determinar por que o processamento não teve sucesso.

Criar uma dead letter queue

Uma dead letter queue é criada da mesma maneira que uma fila regular, mas tem as seguintes restrições:

- Uma dead letter queue deve ter o mesmo tipo de fila (FIFO ou padrão) da fila de origem.
- Uma fila de mensagens não entregues deve ser criada usando-se a mesma Conta da AWS e região da fila de origem.

Criamos aqui duas filas do Amazon SQS idênticas, uma que funcionará como a dead letter queue:

Importações

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

Código da

```
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;
    }
}
```

Veja o exemplo completo no GitHub.

Designar uma dead letter queue para uma fila de origem

Para designar uma fila de mensagens mortas, é necessário criar primeiro uma política de redirecionamento e definir a política nos atributos da fila. Uma política de redirecionamento é especificada em JSON e determina o ARN da fila de mensagens mortas, além do número máximo de vezes em que a mensagem pode ser recebida e não processada antes ser enviada para a fila de mensagens mortas.

Para definir a política de redirecionamento para a fila de origem, chame o método `setQueueAttributes` da classe do AmazonSQS com um objeto SetQueueAttributesRequest para o qual você definiu o atributo `RedrivePolicy` com a política de redirecionamento JSON.

Importações

```
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Código da

```
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);
```

Veja o exemplo completo no GitHub.

Mais informações

- Usar filas de mensagens não entregues doAmazon SQS no Guia do desenvolvedor do Amazon SQS
- SetQueueAttributes na Referência de API do Amazon SQS

Amazon SWF Exemplos de usando a AWS SDK for Java

OAmazon SWF é um serviço de gerenciamento de fluxo de trabalho que ajuda desenvolvedores a compilar e escalar fluxos de trabalho distribuídos que tenham etapas paralelas ou sequenciais que consistem em atividades, fluxos de trabalho filho ou até mesmo tarefas Lambda.

Existem duas maneiras de trabalhar com o Amazon SWF usando o AWS SDK for Java: com o objeto client do SWF ou usando o AWS Flow Framework para Java. O AWS Flow Framework para Java é mais difícil de configurar inicialmente, porque ele usa muitas anotações e conta com bibliotecas

adicionais, como AspectJ e Spring Framework. No entanto, para projetos grandes ou complexos, você economizará tempo de codificação usando o AWS Flow Framework para Java. Para obter mais informações, consulte o Guia do desenvolvedor do AWS Flow Framework para Java.

Esta seção apresenta exemplos de como programar o Amazon SWF usando o cliente do AWS SDK for Java diretamente.

Tópicos

Conceitos básicos do SWF

Esses são padrões gerais para trabalhar com o Amazon SWF usando o AWS SDK for Java. Ele foi desenvolvido principalmente para referência. Para obter um tutorial introdutório mais completo, consulte Compilar um aplicativo do Amazon SWF simples.

Dependências

Os aplicativos do Amazon SWF básicos exigem as seguintes dependências, incluídas com o AWS SDK for Java:

- aws-java-sdk-1.12.*.jar
- commons-logging-1.2.*.jar
- httpclient-4.3.*.jar
- httpcore-4.3.*.jar
- jackson-annotations-2.12.*.jar
- jackson-core-2.12.*.jar
- jackson-databind-2.12.*.jar
- joda-time-2.8.*.jar

> **Note**
>
> Os números da versão desses pacotes serão diferentes dependendo da versão do SDK que você tiver, mas as versões fornecidas com o SDK foram testadas em termos de compatibilidade e são as que deve usar.

Os aplicativos do AWS Flow Framework para Java exigem configuração adicional e dependências adicionais. Consulte o Guia do desenvolvedor do AWS Flow Framework para Java para obter mais informações sobre como usar a estrutura.

Importações

Em geral, você pode usar as seguintes importações no desenvolvimento de código:

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Porém, é uma boa prática importar somente as classes necessárias. Você provavelmente acabará especificando determinadas classes no espaço de trabalho `com.amazonaws.services.simpleworkflow.model`:

```
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 estiver usando o AWS Flow Framework para Java, você importará classes do espaço de trabalho `com.amazonaws.services.simpleworkflow.flow`. Por exemplo:

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

> **Note**
>
> O AWS Flow Framework para Java tem requisitos adicionais além dos AWS SDK for Java base. Para obter mais informações, consulte o Guia do desenvolvedor do AWS Flow Framework para Java.

Usar a classe de cliente do SWF

A interface básica do Amazon SWF é por meio das classes AmazonSimpleWorkflowClient ou AmazonSimpleWorkflowAsyncClient. A principal diferença entre elas é que a classe `\*AsyncClient` retorna objetos Future para programação simultânea (assíncrona).

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

Compilar um aplicativo do Amazon SWF simples

Este tópico apresentará você à programação de aplicativos do Amazon SWF com o AWS SDK for Java, ao mesmo tempo em que apresenta alguns conceitos importantes.

Sobre o exemplo

O projeto de exemplo criará um fluxo de trabalho com uma única atividade que aceita dados do fluxo de trabalho passados pela Nuvem AWS (na tradição de HelloWorld, ele será o nome de alguém a ser cumprimentado) e imprimir um cumprimento em resposta.

Embora isso pareça muito simples superficialmente, os aplicativos do Amazon SWF consistem em várias partes funcionando juntas:

- Um domínio, usado como um contêiner lógico para os dados de execução do fluxo de trabalho.
- Um ou mais fluxos de trabalho que representam os componentes de código que definem a ordem lógica de execução das atividades do fluxo de trabalho e dos fluxos de trabalho filhos.
- Um operador de fluxo de trabalho, também conhecido como administrador, que faça uma sondagem de tarefas de decisão e programe atividades ou fluxos de trabalho filhos em resposta.
- Uma ou mais atividades, cada uma delas representando uma unidade de trabalho no fluxo de trabalho.
- Um operador de atividade que faz uma sondagem de tarefas de atividade e executa métodos de atividade em resposta.
- Uma ou mais listas de tarefas, que são filas mantidas pelo Amazon SWF usadas a fim de emitir requisições para os operadores de fluxo de trabalho e atividade. As tarefas em uma lista indicadas para operadores de fluxo de trabalho são chamadas de tarefas de decisão. As destinadas a operadores de atividade são chamadas de tarefas de atividade.
- Um início de fluxo de trabalho que inicia a execução do fluxo de trabalho.

Nos bastidores, o Amazon SWF orquestra a operação desses componentes, coordenando o fluxo da Nuvem AWS, passando dados entre eles, processando tempos limite e notificações de pulsação, além de registrar em log o histórico de execuções do fluxo de trabalho.

Pré-requisitos

Ambiente de desenvolvimento

O ambiente de desenvolvimento usado neste tutorial consiste em:

- O AWS SDK for Java.
- Apache Maven (3.3.1).
- JDK 1.7 ou posterior. Este tutorial foi desenvolvido e testado usando-se o JDK 1.8.0.
- Um bom editor de textos do Java (sua escolha).

> **Note**
>
> Se usar um sistema de compilação diferente de Maven, você ainda poderá criar um projeto usando as etapas apropriadas ao ambiente e usar os conceitos fornecidos aqui para acompanhar. Mais informações sobre como configurar e usar o AWS SDK for Java com diversos sistemas de compilação são fornecidas em Conceitos básicos.
> Da mesma forma, mas com mais esforço, as etapas mostradas aqui podem ser implementadas usando-se qualquer um dos SDKs da AWS compatíveis com o Amazon SWF.

Todas as dependências externas necessárias estão incluídas com o AWS SDK for Java. Dessa maneira, não há mais nada para fazer download.

Acesso da AWS

Para trabalhar com sucesso neste tutorial, você deve ter acesso ao portal de acesso da AWS, conforme descrito na seção de configuração básica deste guia.

As instruções descrevem como acessar as credenciais temporárias que você copia e cola no arquivo compartilhado local de `credentials`. As credenciais temporárias que você cola devem estar associadas a um perfil do IAM no Centro de Identidade do AWS IAM que tenha permissões para acessar o Amazon SWF. Depois de colar as credenciais temporárias, o arquivo `credentials` será semelhante ao seguinte.

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

Essas credenciais temporárias estão associadas ao perfil `default`.

Criar um projeto SWF

1. Inicie um novo projeto com o Maven:

   ```
   mvn archetype:generate -DartifactId=helloswf \
   -DgroupId=aws.example.helloswf -DinteractiveMode=false
   ```

   Isso criará um novo projeto com uma estrutura de projeto maven padrão:

   ```
   helloswf
   ### pom.xml
   ### src
       ### main
       #   ### java
       #       ### aws
       #           ### example
       #               ### helloswf
       #                   ### App.java
       ### test
           ### ...
   ```

   É possível ignorar ou excluir o diretório `test` e tudo o que ele contiver, e não usaremos isso neste tutorial. Também é possível excluir `App.java`, porque o substituiremos por novas classes.

2. Edite o arquivo `pom.xml` do projeto e adicione o módulo aws-java-sdk-simpleworkflow incluindo uma dependência para ele dentro do bloco `<dependencies>`.

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

3. Verifique se o Maven compila o projeto com suporte ao JDK 1.7+. Adicione o seguinte ao projeto (antes ou depois do bloco `<dependencies>`) em `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>
```

Codificar o projeto

O projeto de exemplo consistirá em quatro aplicativos separados, que analisaremos um por um:

- HelloTypes.java: contém o domínio do projeto, a atividade e os dados do tipo de fluxo de trabalho, compartilhados com os outros componentes. Ele também processa como registrar esses tipos com SWF.
- ActivityWorker.java: contém o operador de atividade, que faz uma sondagem de tarefas de atividade e executa atividades em resposta.
- WorkflowWorker.java: contém o operador de fluxo de trabalho (administrador), que faz uma sondagem de tarefas de administração e programa novas atividades.
- WorkflowStarter.java: contém o início do fluxo de trabalho, que inicia uma nova execução de fluxo de trabalho, o que fará o SWF começar a gerar tarefas de decisão e fluxo de trabalho para consumo pelos operadores.

Etapas comuns a todos os arquivos de origem

Todos os arquivos criados por você para hospedar as classes do Java terão algumas coisas em comum. Pensando no tempo, essas etapas serão implícitas sempre que você adicionar um novo arquivo ao projeto:

1. Crie o arquivo no diretório `src/main/java/aws/example/helloswf/` do projeto.

2. Adicione uma declaração `package` ao início de cada arquivo para declarar o namespace. O projeto de exemplo usa:

```
package aws.example.helloswf;
```

3. Adicione declarações `import` para a classe AmazonSimpleWorkflowClient e para várias classes no namespace `com.amazonaws.services.simpleworkflow.model`. Para simplificar, usaremos:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Registrar um domínio, tipos de fluxo de trabalho e de atividade

Começaremos criando uma classe executável, `HelloTypes.java`. Este arquivo conterá dados compartilhados que partes diferentes do fluxo de trabalho precisarão conhecer, como o nome e a versão dos tipos de atividade e de fluxo de trabalho, o nome de domínio e o nome da lista de tarefas.

1. Abra o editor de textos e crie o arquivo `HelloTypes.java`, adicionando uma declaração de pacote e importações de acordo com as etapas comuns.
2. Declare a classe `HelloTypes` e atribua a ela valores a serem usados nos tipos de atividade e fluxo de trabalho registrados:

```
    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";
```

   Esses valores serão usados em todo o código.

3. Após as declarações String, crie uma instância da classe AmazonSimpleWorkflowClient. Trata-se da interface básica para os métodos do Amazon SWF fornecidos pelo AWS SDK for Java.

```
private static final AmazonSimpleWorkflow swf =
```

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

O trecho anterior pressupõe que as credenciais temporárias estejam associadas ao perfil `default`. Se você usar um perfil diferente, modifique o código acima da seguinte forma e substitua *`profile_name`* pelo verdadeiro nome do perfil.

```
private static final AmazonSimpleWorkflow swf =
        AmazonSimpleWorkflowClientBuilder
                .standard()
                .withCredentials(new ProfileCredentialsProvider("profile_name"))
                .withRegion(Regions.DEFAULT_REGION)
                .build();
```

4. Adicione uma nova função para registrar um domínio do SWF. Domínio é um contêiner lógico para uma série de tipos de atividade e de fluxo de trabalho do SWF relacionados. Os componentes do SWF só poderão se comunicar entre si se estiverem no mesmo domínio.

```
    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!");
    }
```

Ao registrar um domínio, você dá a ele um nome (qualquer conjunto de 1 a 256 caracteres, exceto `:`, `/`, `|`, caracteres de controle ou a string literal “arn”) e um período de retenção, que é o número de dias por que o Amazon SWF manterá os dados do histórico de execução do fluxo de trabalho depois que uma execução do fluxo de trabalho tiver sido concluída. O período de retenção da execução do fluxo de trabalho máximo é 90 dias. Consulte RegisterDomainRequest para obter informações.

Se já houver um domínio com esse nome, um DomainAlreadyExistsException será lançado. Como não estamos preocupados se o domínio já foi criado, podemos ignorar a exceção.

> **Note**
>
> Esse código demonstra um padrão comum durante o trabalho com métodos do AWS SDK for Java, e os dados do método são fornecidos por uma classe no namespace `simpleworkflow.model`, que você instancia e preenche usando os métodos `0with*` em cadeia.

5. Adicione uma função para registrar um novo tipo de atividade. Uma atividade representa uma unidade de trabalho no fluxo de trabalho.

```
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!");
}
```

Um tipo de atividade é identificado por um nome e uma versão, usados para identificar com exclusividade a atividade em quaisquer outros no domínio em que esteja registrado. As atividades também contêm alguns parâmetros opcionais, como a task-list padrão usada para receber tarefas e dados do SWF e alguns tempos limite diferentes que podem ser usados por você para impor restrições ao tempo em que partes diferentes da execução da atividade podem demorar. Consulte RegisterActivityTypeRequest para obter mais informações.

> **Note**
>
> Todos os valores de tempo limite estão especificados em segundos. Consulte Tipos de tempo limite do Amazon SWF para obter uma descrição completa de como tempos limite afetam as execuções de fluxo de trabalho.

Se o tipo de atividade que você estiver tentando registrar já existir, um TypeAlreadyExistsException será lançado. Adicione uma função para registrar um novo tipo de fluxo de trabalho. Um fluxo de trabalho, também conhecido como um administrador, representa a lógica de execução do fluxo de trabalho.

+

```
    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!");
    }
```

+

Semelhantes a tipos de atividade, os tipos de fluxo de trabalho são identificados por um nome e uma versão, além de ter tempos limite configuráveis. Consulte RegisterWorkflowTypeRequest para obter mais informações.

+

Se o tipo de fluxo de trabalho que você estiver tentando registrar já existir, um TypeAlreadyExistsException será lançado. Por fim, torne a classe executável fornecendo a ela um método `main`, que registrará o domínio, o tipo de atividade e o tipo de fluxo de trabalho por vez:

+

```
    registerDomain();
    registerWorkflowType();
    registerActivityType();
```

Você já pode compilar e executar o aplicativo para executar o script de registro ou continuar codificando os operadores de atividade e de fluxo de trabalho. Assim que o domínio, o fluxo de

trabalho e a atividade tiverem sido registrados, você não precisará reexecutá-los. Esses tipos persistirão até você torná-los obsoletos por conta própria.

Implementar o operador de atividade

Uma atividade é a unidade de trabalho básica em um fluxo de trabalho. Um fluxo de trabalho fornece a lógica, programando atividades a serem executadas (ou outras ações a serem tomadas) em resposta a tarefas de decisão. Um fluxo de trabalho típico normalmente consiste em várias atividades que podem ser executadas de maneira síncrona, assíncrona ou uma combinação de ambas.

O operador de atividade é o bit de código que faz uma sondagem de tarefas de atividade geradas pelo Amazon SWF em resposta a decisões de fluxo de trabalho. Ao receber uma tarefa de atividade, ele executa a atividade correspondente e retorna uma resposta de êxito/falha para o fluxo de trabalho.

Implementaremos um operador de atividade simples que realiza uma única atividade.

1. Abra o editor de textos e crie o arquivo `ActivityWorker.java`, adicionando uma declaração de pacote e importações de acordo com as etapas comuns.

   ```
   import com.amazonaws.regions.Regions;
   import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
   import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
   import com.amazonaws.services.simpleworkflow.model.*;
   ```

2. Adicione a classe `ActivityWorker` ao arquivo e atribua a ele um membro de dados para manter um cliente do SWF que usaremos para interagir com o Amazon SWF:

   ```
       private static final AmazonSimpleWorkflow swf =

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

3. Adicione o método que usaremos como uma atividade:

   ```
   private static String sayHello(String input) throws Throwable {
       return "Hello, " + input + "!";
   }
   ```

   A atividade simplesmente utiliza uma string, integra a um cumprimento e retorna o resultado. Embora não seja muito provável que essa atividade crie uma exceção, é uma boa ideia criar atividades que possam lançar um erro se algo der errado.

4. Adicione um método `main` que usaremos como o método de sondagem da tarefa de atividade. Nós o iniciaremos adicionando alguns códigos para fazer uma sondagem à lista de tarefas para tarefas de atividade:

```
        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();
```

   A atividade recebe tarefas do Amazon SWF chamando o método `pollForActivityTask` do cliente do SWF, especificando o domínio e a lista de tarefas a serem usados no PollForActivityTaskRequest passado.

   Assim que uma tarefa for recebida, recuperaremos um identificador exclusivo para ela, chamando o método `getTaskToken` da tarefa.

5. Em seguida, escreva um código para processar as tarefas recebidas. Adicione o seguinte ao método `main`, logo depois do código que faz uma sondagem à tarefa e recupera o token da tarefa.

```
    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 o token da tarefa não for `null`, poderemos começar executando o método de atividade (`sayHello`), fornecendo a ele os dados de entrada enviados com a tarefa.

Se a tarefa for bem-sucedida (sem erro gerado), o operador responderá ao SWF chamando o método `respondActivityTaskCompleted` do cliente do SWF com um objeto RespondActivityTaskCompletedRequest que contém o token da tarefa e os dados resultantes da atividade.

Por outro lado, se a tarefa tiver falhado, responderemos chamando o método `respondActivityTaskFailed` com um objeto RespondActivityTaskFailedRequest, passando a ela o token da tarefas e as informações sobre o erro.

> **Note**
>
> Essa atividade não será desligada de maneira tranquila, se eliminada. Embora esteja além do escopo deste tutorial, uma implementação alternativa desse operador de atividade é apresentada no tópico complementar, Desligar operadores de atividade e fluxo de trabalho de maneira tranquila.

Implementar o operador de fluxo de trabalho

A lógica do fluxo de trabalho reside em um código conhecido como um operador de fluxo de trabalho. O operador de fluxo de trabalho faz uma sondagem para tarefas de decisão enviadas pelo Amazon SWF no domínio e na lista de tarefas padrão com que o tipo de fluxo de trabalho foi registrado.

Quando recebe uma tarefa, o operador de fluxo de trabalho toma algum tipo de decisão (normalmente se deve programar uma nova atividade ou não) e utiliza uma ação apropriada (como programar a atividade).

1. Abra o editor de textos e crie o arquivo `WorkflowWorker.java`, adicionando uma declaração de pacote e importações de acordo com as etapas comuns.
2. Adicione algumas importações adicionais ao arquivo:

```
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. Declare a classe `WorkflowWorker` e crie uma instância da classe AmazonSimpleWorkflowClient usada para acessar os métodos SWF.

```
    private static final AmazonSimpleWorkflow swf =

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

4. Adicione o método `main`. O método fica em loop continuamente, fazendo uma sondagem de tarefas de decisão usando o método `pollForDecisionTask` do cliente do SWF. O PollForDecisionTaskRequest apresenta os detalhes.

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

Assim que uma tarefa for recebida, chamaremos o método `getTaskToken`, que retornará uma string que poderá ser usada para identificar a tarefa. Se o token retornado não for `null`, processaremos ainda mais no método `executeDecisionTask`, passando o token da tarefa e a lista de objetos HistoryEvent enviados com a tarefa.

5. Adicione o método `executeDecisionTask`, utilizando o token da tarefa (uma `String`) e a lista `HistoryEvent`.

```
    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;
```

Também configuramos alguns membros de dados para acompanhar coisas como:

- Uma lista de objetos Decisão usados para relatar os resultados do processamento da tarefa.
- Uma string para armazenar a entrada do fluxo de trabalho fornecida pelo evento "WorkflowExecutionStarted"
- uma contagem das atividades programadas e abertas (em execução) para evitar programar a mesma atividade quando ela já tiver sido programada ou estiver em execução no momento.
- um booliano para indicar que a atividade foi concluída.
- Uma string para manter os resultados da atividade, a fim de retorná-los como o resultado do fluxo de trabalho.

6. Em seguida, adicione um código a `executeDecisionTask` para processar os objetos `HistoryEvent` que foram enviados com a tarefa, com base no tipo de evento informado pelo método `getEventType`.

```
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("]");
```

Tendo em vista o fluxo de trabalho, estamos mais interessados:

- no evento "WorkflowExecutionStarted", que indica que a execução do fluxo de trabalho foi iniciada (normalmente isso significa que você deve executar a primeira atividade no fluxo de trabalho) e apresenta a entrada inicial fornecida para o fluxo de trabalho. Nesse caso, trata-se da parte do nome do cumprimento. Por isso, ela é salva em uma string para ser usada durante a programação da atividade a ser executada.

- no evento "ActivityTaskCompleted", enviado assim que a atividade programada é concluída. Os dados do evento também incluem o valor de retorno da atividade concluída. Como temos somente uma atividade, usaremos esse valor como o resultado de todo o fluxo de trabalho.

Os outros tipos de evento poderão ser usados se o fluxo de trabalho precisar deles. Consulte a descrição da classe HistoryEvent para obter informações sobre cada tipo de evento.

+ OBSERVAÇÃO: as strings em instruções `switch` foram introduzidas no Java 7. Se estiver usando uma versão anterior do Java, será possível usar a classe EventType para converter a `String` retornado por `history_event.getType()` em um valor enum e novamente em uma `String`, se necessário:

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

1. Depois da instrução `switch`, adicione mais código para responder com uma decisão apropriada com base na tarefa que foi recebida.

```
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 a atividade ainda não tiver sido programada, responderemos com uma decisão `ScheduleActivityTask`, que fornecerá informações em uma estrutura ScheduleActivityTaskDecisionAttributes sobre a atividade que o Amazon SWF deve programar em seguida, incluindo também todos os dados que o Amazon SWF deve enviar para a atividade.
- Se a atividade tiver sido concluída, vamos considerar todo o fluxo de trabalho concluído e responder com uma decisão `CompletedWorkflowExecution`, preenchendo uma estrutura CompleteWorkflowExecutionDecisionAttributes para fornecer detalhes sobre o fluxo de trabalho concluído. Neste caso, retornamos o resultado da atividade.

Em qualquer um dos casos, as informações sobre a decisão são adicionadas à lista `Decision` que foi declarada na parte superior do método.

2. Conclua a tarefa de decisão retornando a lista de objetos `Decision` coletados durante o processamento da tarefa. Adicione esse código ao final do método `executeDecisionTask` que estávamos escrevendo:

```
swf.respondDecisionTaskCompleted(
    new RespondDecisionTaskCompletedRequest()
        .withTaskToken(taskToken)
        .withDecisions(decisions));
```

O método `respondDecisionTaskCompleted` do cliente do SWF utiliza o token da tarefa que identifica a tarefa, bem como a lista de objetos `Decision`.

Implementar o início do fluxo de trabalho

Por fim, escreveremos um código para iniciar a execução do fluxo de trabalho.

1. Abra o editor de textos e crie o arquivo `WorkflowStarter.java`, adicionando uma declaração de pacote e importações de acordo com as etapas comuns.

2. Adicione a classe `WorkflowStarter`:

```
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() + "'.");
    }
}
```

A classe `WorkflowStarter` consiste em um único método `main`, que utiliza um argumento opcional passado na linha de comando como dados de entrada para o fluxo de trabalho.

O método de cliente do SWF, `startWorkflowExecution`, utiliza um objeto StartWorkflowExecutionRequest como entrada. Aqui, além de especificar o domínio e o tipo de fluxo de trabalho para execução, fornecemos:

- um nome de execução do fluxo de trabalho legível por humanos
- dados de entrada do fluxo de trabalho (fornecidos na linha de comando no exemplo)
- um valor de tempo limite que representa por quanto tempo, em segundos, todo o fluxo de trabalho deve ser executado.

O objeto Executar que `startWorkflowExecution` retorna fornece um ID de execução, um valor que pode ser usado para identificar a execução desse fluxo de trabalho em especial no histórico do Amazon SWF das execuções de fluxo de trabalho.

+ OBSERVAÇÃO: o ID de execução é gerado pelo Amazon SWF e não tem o mesmo nome de execução do fluxo de trabalho passado por você ao iniciar a execução do fluxo de trabalho.

Compilar o exemplo

Para compilar o projeto de exemplo com o Maven, vá até o diretório `helloswf` e digite:

```
mvn package
```

O `helloswf-1.0.jar` resultante será gerado no diretório `target`.

Executar o exemplo

O exemplo consiste em quatro classes executáveis separadas, executadas de maneira independente entre si.

> **Note**
>
> Se estiver usando um sistema Linux, macOS ou Unix, você poderá executar todas, uma depois da outra, em uma única janela do terminal. Se estiver executando o Windows, você deverá abrir duas instâncias de linha de comando adicionais e navegar até o diretório `helloswf` em cada uma delas.

Configurar o classpath do Java

Embora o Maven tenha processado as dependências para você, para executar o exemplo, será necessário fornecer a biblioteca de SDK da AWS e as dependências no classpath do Java. Você pode definir a variável de ambiente `CLASSPATH` como o local das bibliotecas do SDK da AWS e o diretório `third-party/lib` no SDK, que inclui as dependências necessárias:

```
export CLASSPATH='target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/
lib/*'
java example.swf.hello.HelloTypes
```

ou usar a opção `-cp` do comando **`java`** para definir o classpath, ao mesmo tempo em que executa todos os aplicativos.

```
java -cp target/helloswf-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/* \
  example.swf.hello.HelloTypes
```

O estilo usado cabe a você. Se você não teve problemas ao compilar o código, tentou executar os exemplos e obteve uma série de erros "NoClassDefFound", provavelmente será porque o classpath estava definido de maneira incorreta.

Registrar o domínio, tipos de fluxo de trabalho e de atividade

Para executar os operadores e o início do fluxo de trabalho, será necessário registrar o domínio e os tipos de fluxo de trabalho e de atividade. O código para fazer isso foi implementado em Registrar um fluxo de trabalho de domínio e tipos de atividade.

Depois da criação, se você definiu o CLASSPATH, será possível executar o código de registro executando o comando:

```
    echo 'Supply the name of one of the example classes as an argument.'
```

Iniciar os operadores de atividade e de fluxo de trabalho

Agora que os tipos foram registrados, você poderá iniciar os operadores de atividade e de fluxo de trabalho. Eles continuarão sendo executados e sondando tarefas até serem eliminados. Dessa maneira, é necessário executá-los em janelas de terminal separadas ou, se estiver executando no Linux, no macOS ou no Unix, será possível usar o operador & para fazer cada um deles gerar um processo separado quando executado.

```
    echo 'If there are arguments to the class, put them in quotes after the class
name.'
    exit 1
```

Se você estiver executando esses comandos em janelas separadas, omita o operador & final de cada linha.

Iniciar a execução de fluxo de trabalho

Agora que os operadores de atividade e de fluxo de trabalho estão fazendo uma sondagem, você pode começar a execução do fluxo de trabalho. Esse processo será executado até o fluxo de trabalho retornar um status concluído. Você deve executá-lo em uma nova janela do terminal (a menos que tenha executado os operadores como novos processos gerados usando o operador &).

```
fi
```

> **Note**
>
> Se você quiser fornecer os próprios dados de entrada, que serão passados primeiro para o fluxo de trabalho e, em seguida, para a atividade, adicione-os à linha de comando. Por exemplo:
>
> ```
> echo "## Running $className..."
> ```

Assim que começar a execução do fluxo de trabalho, você deverá começar a ver a saída entregue por ambos os operadores e pela própria execução do fluxo de trabalho. Quando o fluxo de trabalho for finalmente concluído, a saída será impressa na tela.

Fonte completa deste exemplo

Você pode procurar a origem completa desse exemplo no Github no repositório aws-java-developer-guide.

Para obter mais informações

- Os operadores apresentados aqui poderão resultar em tarefas perdidas, se forem desligados enquanto a sondagem de um fluxo de trabalho estiver acontecendo. Para saber como desligar

operadores de maneira tranquila, consulte Desligar operadores de atividade e fluxo de trabalho de maneira tranquila.

- Para saber mais sobre o Amazon SWF, visite a página inicial do Amazon SWF ou consulte o Guia do desenvolvedor do Amazon SWF.
- Você pode usar o AWS Flow Framework para Java para escrever fluxos de trabalho mais complexos em um estilo Java elegante usando anotações. Para saber mais, consulte o Guia do desenvolvedor do AWS Flow Framework para Java.

LambdaTarefas do

Como alternativa às atividades do Amazon SWF ou em conjunto com elas, você pode usar funções Lambda para representar unidades de trabalho nos fluxos de trabalho e programá-las de maneira semelhante a atividades.

Este tópico descreve como implementar tarefas Lambda do Amazon SWF usando o AWS SDK for Java. Para obter mais informações sobre tarefas do Lambda em geral, consulte Tarefas do AWS Lambda em Guia do desenvolvedor do Amazon SWF.

Configurar um perfil do IAM entre serviços para executar a função Lambda

Para o Amazon SWF executar a função do Lambda, você precisa configurar um perfil do IAM para dar ao Amazon SWF permissão para executar funções do Lambda em seu nome. Para obter informações completas sobre como fazer isso, consulte Tarefas do AWS Lambda.

Será necessário o nome do recurso da Amazon (ARN) desse perfil do IAM ao registrar um fluxo de trabalho que usará tarefas do Lambda.

Criar uma função do Lambda

Você pode escrever funções Lambda em várias linguagens diferentes, inclusive Java. Para obter informações completas sobre como criar, implantar e usar funções do Lambda, consulte o Guia do desenvolvedor do AWS Lambda.

> **Note**
>
> Não importa a linguagem usada para escrever a função do Lambda, pois ela pode ser programada e executada por qualquer fluxo de trabalho do Amazon SWF, independentemente da linguagem na qual o código de fluxo de trabalho foi escrito. O

Amazon SWF processa os detalhes de como executar a função e passar os dados de e para ele.

Veja a seguir uma função do Lambda simples que pode ser usada no lugar da atividade em Criar um aplicativo do Amazon SWF simples.

- Essa versão foi escrita em JavaScript, que pode ser inserido diretamente usando-se o Console de gerenciamento da AWS:

```
exports.handler = function(event, context) {
    context.succeed("Hello, " + event.who + "!");
};
```

- Aqui está a mesma função escrita em Java, que você também pode implantar e executar no 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**
>
> Para saber mais sobre como implantar funções do Java no Lambda, consulte Criar um pacote de implantação (Java) no Guia do desenvolvedor do AWS Lambda. Você também deve observar a seção intitulada Programar modelo para criar funções do Lambda no Java.

As funções do Lambda utilizam um objeto event ou input como o primeiro parâmetro e um objeto context como o segundo, o que fornece informações sobre a solicitação para executar a função do Lambda. Essa função em especial espera que a entrada esteja em JSON, com um campo who definido como o nome usado para criar o cumprimento.

Registrar um fluxo de trabalho a ser usado com o Lambda

Para um fluxo de trabalho programar uma função do Lambda, você deve fornecer o nome do perfil do IAM que dá ao Amazon SWF permissão para invocar funções Lambda. Você pode definir isso durante o registro do fluxo de trabalho usando os métodos `withDefaultLambdaRole` ou `setDefaultLambdaRole` de RegisterWorkflowTypeRequest.

```
System.out.println("** Registering the workflow type '" + WORKFLOW + "-" +
 WORKFLOW_VERSION
        + "'.");
try {
    swf.registerWorkflowType(new RegisterWorkflowTypeRequest()
        .withDomain(DOMAIN)
        .withName(WORKFLOW)
        .withDefaultLambdaRole(lambda_role_arn)
        .withVersion(WORKFLOW_VERSION)
        .withDefaultChildPolicy(ChildPolicy.TERMINATE)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskStartToCloseTimeout("30"));
}
catch (TypeAlreadyExistsException e) {
```

Programar uma tarefa do Lambda

Programar uma tarefa do Lambda é semelhante a programar uma atividade. Você fornece uma Decisão com um DecisionType “ScheduleLambdaFunction” e com ScheduleLambdaFunctionDecisionAttributes.

```
running_functions == 0 && scheduled_functions == 0) {
AWSLambda lam = AWSLambdaClientBuilder.defaultClient();
GetFunctionConfigurationResult function_config =
    lam.getFunctionConfiguration(
            new GetFunctionConfigurationRequest()
                .withFunctionName("HelloFunction"));
String function_arn = function_config.getFunctionArn();

ScheduleLambdaFunctionDecisionAttributes attrs =
    new ScheduleLambdaFunctionDecisionAttributes()
        .withId("HelloFunction (Lambda task example)")
        .withName(function_arn)
        .withInput(workflow_input);

decisions.add(
```

Em `ScheduleLambdaFuntionDecisionAttributes`, você deve fornecer um nome, que é o ARN da função do Lambda a ser chamada, e um id, que é o nome que o Amazon SWF usará para identificar a função do Lambda em logs do histórico.

Você também pode fornecer input opcional para a função do Lambda e definir o valor começar para encerrar tempo limte, que é o número de segundos em que a função do Lambda tem permissão para ser executada antes de gerar um evento `LambdaFunctionTimedOut`.

> **Note**
>
> Esse código usa o AWSLambdaClient para recuperar o ARN da função do Lambda, dado o nome da função. Você pode usar essa técnica para evitar realizar a codificação rígida do ARN completo (o que inclui o ID de Conta da AWS) no código.

Processar eventos de função do Lambda no administrador

As tarefas do Lambda vão gerar vários eventos em que você pode atuar ao fazer uma sondagem de tarefas de decisão no operador de fluxo de trabalho, correspondente ao ciclo de vida da tarefa do

Lambda, com valores EventType como `LambdaFunctionScheduled`, `LambdaFunctionStarted` e `LambdaFunctionCompleted`. Se a função do Lambda falhar ou demorar mais do que o valor de tempo limite definido, você receberá um tipo de evento `LambdaFunctionFailed` ou `LambdaFunctionTimedOut`, respectivamente.

```
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;
```

Receber a saída da função do Lambda

Você pode receber um `LambdaFunctionCompleted`EventType, you can retrieve your 0 function's return value by first calling `getLambdaFunctionCompletedEventAttributes no HistoryEvent para obter um objeto LambdaFunctionCompletedEventAttributes e chamar o método `getResult` para recuperar a saída da função do Lambda:

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

Fonte completa deste exemplo

Você pode procurar a origem completa :github:`<awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf_lambda/> desse exemplo no Github no repositório aws-java-developer-guide.

Desligar operadores de atividade e de fluxo de trabalho de maneira tranquila

O tópico Compilar um aplicativo Amazon SWF simples apresentou uma implementação completa de um aplicativo de fluxo de trabalho simples que consiste em um aplicativo de registro, um operador de atividade e de fluxo de trabalho, além de um início de fluxo de trabalho.

As classes de operador foram projetadas para serem executadas continuamente, sondando tarefas enviadas pelo Amazon SWF para executar atividades ou retornar decisões. Assim que uma requisição de sondagem for feita, o Amazon SWF vai registrar quem fez a sondagem e tentar atribuir uma tarefa.

Se o operador de fluxo de trabalho for encerrado durante uma sondagem longa, o Amazon SWF ainda poderá tentar enviar uma tarefa para o operador encerrado, resultando em uma tarefa perdida (até a tarefa expirar).

Uma maneira de processar essa situação é aguardar todas as requisições de sondagem longa retornarem antes do operador terminar.

Neste tópico, vamos reescrever o operador de atividade de `helloswf`, usando ganchos de desligamento do Java para fazer um desligamento normal do operador de atividade.

Aqui está o código 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()));
                }
            }
        }
    }
}
```

Nesta versão, o código de sondagem que estava na função `main` na versão original foi migrado para o próprio método, `pollAndExecute`.

Agora a função `main` usa um CountDownLatch com um hook de desligamento para fazer o thread aguardar até 60 segundos após o término ser solicitado antes de permitir o desligamento do thread.

Registro de domínios

Cada fluxo de trabalho e atividade no Amazon SWF precisa de um domínio para execução.

1. Crie um novo objeto RegisterDomainRequest, fornecendo pelo menos o nome de domínio e o período de retenção da execução do fluxo de trabalho (ambos os parâmetros são obrigatórios).
2. Chame o método AmazonSimpleWorkflowClient.registerDomain com o objeto RegisterDomainRequest.
3. Intercepte DomainAlreadyExistsException se o domínio que você estiver solicitando já existir (nesse caso, geralmente nenhuma ação é necessária).

O código a seguir demonstra este procedimento:

```
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!");
    }
```

```
}
```

Listar domínios

Você pode listar os domínios do Amazon SWF associados à conta e à região da AWS por tipo de registro.

1. Crie um objeto ListDomainsRequest e especifique o status de registro dos domínios nos quais você tenha interesse. Isso é obrigatório.
2. Chame AmazonSimpleWorkflowClient.listDomains com o objeto ListDomainRequest. Os resultados são fornecidos em um objeto DomainInfos.
3. Chame getDomainInfos no objeto retornado para obter uma lista de objetos DomainInfo.
4. Chame getName em cada objeto DomainInfo para obter o nome.

O código a seguir demonstra este procedimento:

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

Amostras de código incluídas no SDK

O AWS SDK for Java acompanha exemplos de código que demonstram muitos dos recursos do SDK nos programas executáveis, compiláveis. Você pode estudar ou modificá-los para implementar as próprias soluções da AWS usando o AWS SDK for Java.

Como obter os exemplos

Os exemplos de código do AWS SDK for Java são fornecidos no diretório de exemplos do SDK. Se tiver obtido por download e instalado o SDK usando as informações em Configurar o AWS SDK for Java, você já terá os exemplos no sistema.

Você também pode visualizar os exemplos mais recentes no repositório GitHub do AWS SDK for Java, no diretório src/samples.

Compilar e executar os exemplos usando a linha de comando

Entre os exemplos estão scripts de compilação Ant, de maneira que você possa compilar e executá-los facilmente na linha de comando. Todo exemplo também contém um arquivo README em formato HTML com informações específicas de cada exemplo.

> **Note**
>
> Se você estiver procurando o código de exemplo no GitHub, clique no botão Raw na exibição do código-fonte ao visualizar o arquivo README.html do exemplo. Em modo bruto, o HTML será renderizado conforme desejado no navegador.

Pré-requisitos

Para executar qualquer um dos exemplos do AWS SDK for Java, você precisa definir as credenciais da AWS no ambiente ou com a AWS CLI, conforme especificado em Configurar credenciais e região da AWS para desenvolvimento. Os exemplos usam a cadeia de fornecedores de credencial padrão sempre que possível. Dessa forma, definindo as credenciais assim, você pode evitar a prática arriscada de inserir as credenciais da AWS em arquivos dentro do diretório de código-fonte (onde o check-in deles pode ser feito inadvertidamente, além de compartilhados publicamente).

Executar os exemplos

1. Mude para o diretório que contém o código de exemplo. Por exemplo, se você está no diretório raiz do download do SDK da AWS e deseja executar o exemplo `AwsConsoleApp`, digite:

   ```
   cd samples/AwsConsoleApp
   ```

2. Compile e execute o exemplo com Ant. O alvo da compilação padrão realiza ambas as ações. Dessa forma, basta você inserir:

```
ant
```

As informações de impressões de exemplo para a saída padrão, por exemplo:

```
===========================================

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.
```

Compilar e executar os exemplos usando o IDE do Eclipse

Se usar o AWS Toolkit for Eclipse, você também poderá iniciar um novo projeto no Eclipse com base no AWS SDK for Java ou adicionar o SDK a um projeto do Java existente.

Pré-requisitos

Após a instalação do AWS Toolkit for Eclipse, recomendamos configurar o Toolkit com as credenciais de segurança. Você pode fazer isso a qualquer momento escolhendo Preferências no menu Janela do Eclipse e a seção Kit de ferramentas da AWS.

Executar os exemplos

1. Abra o Eclipse.
2. Criar um novo projeto da AWS em Java. No Eclipse, no menu File, escolha New e clique em Project. O assistente New Project é aberto.
3. Expanda a categoria AWS e escolha AWS Java Project.
4. Escolha Próximo. A página de configurações do projeto é exibida.
5. Insira um nome na caixa Project Name. O grupo de exemplos do AWS SDK for Java exibe os exemplos disponíveis no SDK, conforme descrito anteriormente.

6. Selecione os exemplos que você deseja incluir no projeto marcando cada caixa de seleção.
7. Insira credenciais de usuário da AWS. Se você já configurou o AWS Toolkit for Eclipse com as credenciais, ele será preenchido automaticamente.
8. Escolha Terminar. O projeto é criado e adicionado ao Project Explorer.
9. Escolha o arquivo de exemplo `.java` que você deseja executar. Por exemplo, para o exemplo do Amazon S3, escolha `S3Sample.java`.
10. Escolha Run no menu Run.
11. Clique com o botão direito do mouse no projeto em Project Explorer, aponte para Build Path e escolha Add Libraries.
12. Escolha AWS SDK para Java, selecione Avançar e siga as instruções na tela restantes.

Segurança para o AWS SDK for Java

A segurança da nuvem na Amazon Web Services (AWS) é a nossa maior prioridade. Como cliente da AWS , você contará com um data center e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança. A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como a Segurança da nuvem e a Segurança na nuvem.

Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa todos os serviços oferecidos na AWS nuvem e fornecer serviços que você possa usar com segurança. Nossa responsabilidade de segurança é a maior prioridade em AWS, e a eficácia de nossa segurança é regularmente testada e verificada por auditores terceirizados como parte dos Programas de AWS Conformidade.

Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você está usando e por outros fatores, incluindo a sensibilidade de seus dados, os requisitos da sua organização e as leis e regulamentos aplicáveis.

Esse AWS produto ou serviço segue o modelo de responsabilidade compartilhada por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do AWS serviço, consulte a página de documentação de segurança do serviço e os AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade.

Tópicos

Proteção de dados no AWS SDK for Java 1.x

O modelo de responsabilidade compartilhada se aplica à proteção de dados neste AWS produto ou serviço. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa toda a AWS nuvem. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Esse conteúdo inclui as tarefas de configuração e gerenciamento de segurança dos serviços da AWS que você usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para obter informações sobre proteção de dados na Europa, consulte a postagem do blog sobre o Modelo de Responsabilidade AWS Compartilhada e o GDPR no Blog AWS de Segurança.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure contas de usuário individuais com AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções AWS de criptografia, com todos os controles de segurança padrão nos AWS serviços.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados pessoais armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para obter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que você nunca coloque informações de identificação confidenciais, como números de conta dos seus clientes, em campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com este AWS produto ou serviço ou outros AWS serviços usando o console, a API ou AWS SDKs. AWS CLI Todos os dados que você inserir neste AWS produto, serviço ou outros serviços podem ser coletados para inclusão nos registros de diagnóstico. Ao fornecer um URL para um servidor externo, não inclua informações de credenciais no URL para validar a solicitação a esse servidor.

AWS SDK for Java suporte para TLS

As informações a seguir se aplicam somente à implementação de Java SSL (a implementação SSL padrão no AWS SDK for Java). Se você estiver usando uma implementação SSL diferente, consulte sua implementação SSL específica para saber como impor versões TLS.

Como verificar a versão do TLS

Consulte a documentação do seu provedor de máquina virtual do Java (JVM) para determinar quais versões de TLS têm suporte em sua plataforma. Para alguns JVMs, o código a seguir imprimirá quais versões de SSL são suportadas.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProto
```

Para ver o handshake SSL em ação, e qual versão do TLS é usada, você pode usar a propriedade do sistema javax.net.debug.

```
java app.jar -Djavax.net.debug=ssl
```

> **Note**
>
> O TLS 1.3 é incompatível com as versões 1.9.5 a 1.10.31 do SDK para Java. Para ter mais informações, consulte a seguinte postagem no blog. https://aws.amazon.com/blogs/desenvolvedor/tls-1-3- - -1-9-5-para-1-10-31/ incompatibility-with-aws-sdk for-java-versions

Aplicar uma versão mínima do TLS

O SDK sempre prefere a versão mais recente do TLS compatível com a plataforma e o serviço. Se você deseja aplicar uma versão mínima específica do TLS, consulte a documentação da sua JVM. Para sistemas baseados em OpenJDK JVMs, você pode usar a propriedade do sistema. `jdk.tls.client.protocols`

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

Consulte a documentação da sua JVM para obter os valores suportados dos PROTOCOLOS.

Gerenciamento de Identidade e Acesso

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar AWS os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz AWS.

Usuário do serviço — Se você Serviços da AWS costuma fazer seu trabalho, seu administrador fornece as credenciais e as permissões de que você precisa. À medida que você usa mais AWS recursos para fazer seu trabalho, talvez precise de permissões adicionais. Entender como o acesso é gerenciado pode ajudar a solicitar as permissões corretas ao administrador. Se você não conseguir acessar um recurso no AWS, consulte Solução de problemas AWS de identidade e acesso o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Administrador de serviços — Se você é responsável pelos AWS recursos da sua empresa, provavelmente tem acesso total AWS a. É seu trabalho determinar quais AWS recursos e recursos seus usuários do serviço devem acessar. Envie as solicitações ao administrador do IAM para alterar as permissões dos usuários de serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM. Para saber mais sobre como sua empresa pode usar o IAM com AWS, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Administrador do IAM: se você for um administrador do IAM, talvez queira saber detalhes sobre como pode gravar políticas para gerenciar o acesso ao AWS. Para ver exemplos de políticas AWS

baseadas em identidade que você pode usar no IAM, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários humanos usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório corporativo, provedor de identidade da web ou Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem funções que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos Centro de Identidade do AWS IAM. Para saber mais, consulte O que é o IAM Identity Center? no Guia do usuário do Centro de Identidade do AWS IAM .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

As funções do IAM são úteis para acesso de usuários federados, permissões temporárias de usuários do IAM, acesso entre contas, acesso entre serviços e aplicativos executados na Amazon. EC2 Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você anexa a uma identidade (usuário, grupo ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — Especifique as permissões máximas para uma organização ou unidade organizacional em AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs) — Defina o máximo de permissões disponíveis para recursos em suas contas. Para obter mais informações, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como Serviços da AWS trabalhar com o IAM

Para ter uma visão de alto nível de como Serviços da AWS funciona com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

Para saber como usar um específico AWS service (Serviço da AWS) com o IAM, consulte a seção de segurança do Guia do usuário do serviço relevante.

Solução de problemas AWS de identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com AWS um IAM.

Tópicos

- Não estou autorizado a realizar uma ação em AWS
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS recursos

Não estou autorizado a realizar uma ação em AWS

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `awes:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso *`my-example-widget`* usando a ação `awes:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não está autorizado a executar a ação `iam:PassRole`, as suas políticas devem ser atualizadas para permitir que você passe uma função para o AWS.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazê-lo, você deve ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta utilizar o console para executar uma ação no AWS. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

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

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS recursos

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se é AWS compatível com esses recursos, consulteComo Serviços da AWS trabalhar com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Validação de conformidade para este AWS produto ou serviço

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. Para obter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte a documentação AWS de segurança.

Esse AWS produto ou serviço segue o modelo de responsabilidade compartilhada por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do AWS serviço, consulte a página de documentação de segurança do serviço e os AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade.

Resiliência para este AWS produto ou serviço

A infraestrutura AWS global é construída em torno Regiões da AWS de zonas de disponibilidade.

Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância.

Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Esse AWS produto ou serviço segue o modelo de responsabilidade compartilhada por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do AWS serviço, consulte a página de documentação de segurança do serviço e os AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade.

Segurança da infraestrutura para este AWS produto ou serviço

Esse AWS produto ou serviço usa serviços gerenciados e, portanto, é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar este AWS Produto ou Serviço pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.

- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Esse AWS produto ou serviço segue o modelo de responsabilidade compartilhada por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do AWS serviço, consulte a página de documentação de segurança do serviço e os AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade.

Amazon S3 Migração do cliente de criptografia

Este tópico mostra como migrar seus aplicativos da versão 1 (V1) do cliente de criptografia () para a versão 2 Amazon Simple Storage Service (Amazon S3 V2) e garantir a disponibilidade do aplicativo durante todo o processo de migração.

Pré-requisitos

Amazon S3 a criptografia do lado do cliente exige o seguinte:

- Java 8 ou posterior instalado em seu ambiente de aplicativos. AWS SDK for Java Funciona com o Oracle Java SE Development Kit e com distribuições do Open Java Development Kit (OpenJDK) Amazon Corretto, como Red Hat OpenJDK e JDK. AdoptOpen
- O pacote Bouncy Castle Crypto. Você pode colocar o arquivo.jar do Bouncy Castle no classpath do ambiente do seu aplicativo ou adicionar uma dependência do `bcprov-ext-jdk15on` do artifactId (com o groupId de `org.bouncycastle`) ao seu arquivo`pom.xml` do Maven.

Visão geral da migração

Essa migração acontece em duas fases:

1. Atualizar os clientes existentes para ler novos formatos. Atualize seu aplicativo para usar a versão 1.11.837 ou posterior do AWS SDK for Java e reimplante o aplicativo. Isso permite

que os Amazon S3 clientes do serviço de criptografia do lado do cliente em seu aplicativo descriptografem objetos criados pelos clientes do serviço V2. Se seu aplicativo usa vários AWS SDKs, você deve atualizar cada SDK separadamente.

2. Migrar clientes de criptografia e descriptografia para a V2. Depois que todos os seus clientes de criptografia V1 puderem ler os formatos de criptografia V2, atualize os Amazon S3 clientes de criptografia e descriptografia do lado do cliente no código do aplicativo para usar seus equivalentes V2.

Atualizar os clientes existentes para ler novos formatos

O cliente de criptografia V2 usa algoritmos de criptografia que as versões mais antigas AWS SDK for Java do não suportam.

A primeira etapa da migração é atualizar seus clientes de criptografia V1 para usar a versão 1.11.837 ou posterior do. AWS SDK for Java(Recomendamos que você atualize para a versão mais recente, que pode ser encontrada na Referência de API do Java versão 1.x.) Para fazer isso, atualize a dependência na configuração do seu projeto. Depois que a configuração do projeto for atualizada, recrie seu projeto e reimplante-o.

Depois de concluir essas etapas, os clientes de criptografia V1 do seu aplicativo poderão ler objetos escritos por clientes de criptografia V2.

Atualizar a dependência na configuração do seu projeto

Modifique o arquivo de configuração do projeto (por exemplo, pom.xml ou build.gradle) para usar a versão 1.11.837 ou posterior do AWS SDK for Java. Em seguida, recrie seu projeto e reimplante-o.

Concluir essa etapa antes de implantar o novo código do aplicativo ajuda a garantir que as operações de criptografia e descriptografia permaneçam consistentes em toda a sua frota durante o processo de migração.

Exemplos usando o Maven

Trecho de um arquivo 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>
```

Exemplo usando o Gradle

Trecho de um arquivo build.gradle:

```
dependencies {
  implementation platform('com.amazonaws:aws-java-sdk-bom:1.11.837')
  implementation 'com.amazonaws:aws-java-sdk-s3'
}
```

Migrar clientes de criptografia e descriptografia para a V2

Depois que seu projeto for atualizado com a versão mais recente do SDK, você poderá modificar o código do aplicativo para usar o cliente V2. Para fazer isso, primeiro atualize seu código para usar o novo criador de clientes de serviço. Em seguida, forneça materiais de criptografia usando um método no construtor que tenha sido renomeado e configure ainda mais seu cliente de serviço conforme necessário.

Esses trechos de código demonstram como usar a criptografia do lado do cliente com o e fornecem comparações entre os AWS SDK for Java clientes de criptografia 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();
```

O exemplo acima define o `cryptoMode` como `AuthenticatedEncryption`. Essa é uma configuração que permite que um cliente de criptografia V2 leia objetos que foram escritos por um cliente de criptografia V1. Se seu cliente não precisar da capacidade de ler objetos escritos por um cliente V1, recomendamos usar a configuração padrão `StrictAuthenticatedEncryption` em vez disso.

Construir um cliente de criptografia V2

O cliente de criptografia V2 pode ser construído chamando o AmazonS3 EncryptionClient v2.encryptionBuilder ().

Você pode substituir todos os seus clientes de criptografia V1 existentes por clientes de criptografia V2. Um cliente de criptografia V2 sempre poderá ler qualquer objeto que tenha sido escrito por um cliente de criptografia V1, desde que você permita que ele faça isso configurando o cliente de criptografia V2 para usar o `. AuthenticatedEncryption `cryptoMode`

A criação de um novo cliente de criptografia V2 é muito semelhante à criação de um cliente de criptografia V1. No entanto, há algumas diferenças:

- Você usará um objeto `CryptoConfigurationV2` para configurar o cliente em vez de um objeto `CryptoConfiguration`. Esse parâmetro é obrigatório.
- A configuração padrão `cryptoMode` para o cliente de criptografia V2 é `StrictAuthenticatedEncryption`. Para o cliente de criptografia V1, é `EncryptionOnly`.
- O método withEncryptionMaterials() no construtor do cliente de criptografia foi renomeado para withEncryptionMaterialsProvider (). Essa é apenas uma mudança cosmética que reflete com mais precisão o tipo de argumento. Você deve usar o novo método ao configurar seu cliente de serviço.

> **Note**
>
> Ao descriptografar com o AES-GCM, leia o objeto inteiro até o fim antes de começar a usar os dados descriptografados. Isso é para verificar se o objeto não foi modificado desde que foi criptografado.

Usar fornecedores de materiais de criptografia

Você pode continuar usando os mesmos provedores de materiais de criptografia e objetos de materiais de criptografia que você já está usando com o cliente de criptografia V1. Essas classes são responsáveis por fornecer as chaves que o cliente de criptografia usa para proteger seus dados. Elas podem ser usadas alternadamente com o cliente de criptografia V2 e V1.

Configurar o cliente de criptografia V2

O cliente de criptografia V2 é configurado com um objeto `CryptoConfigurationV2`. Esse objeto pode ser construído chamando seu construtor padrão e, em seguida, modificando suas propriedades conforme exigido dos padrões.

Os valores padrão para `CryptoConfigurationV2` são:

- `cryptoMode` = `CryptoMode.StrictAuthenticatedEncryption`
- `storageMode` = `CryptoStorageMode.ObjectMetadata`
- `secureRandom` = instância de `SecureRandom`
- `rangeGetMode` = `CryptoRangeGetMode.DISABLED`
- `unsafeUndecryptableObjectPassthrough` = `false`

Observe que não EncryptionOnlyé compatível `cryptoMode` com o cliente de criptografia V2. O cliente de criptografia V2 sempre criptografará o conteúdo usando criptografia autenticada e protegerá as chaves de criptografia de conteúdo (CEKs) usando objetos V2. `KeyWrap`

O exemplo a seguir demonstra como especificar a configuração de criptografia na V1 e como instanciar um objeto V2 para passar para o construtor do cliente de criptografia CryptoConfigurationV2.

V1

```
CryptoConfiguration cryptoConfiguration = new CryptoConfiguration()
        .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

V2

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
        .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

Exemplos adicionais

Os exemplos a seguir demonstram como abordar casos de uso específicos relacionados à migração da V1 para a V2.

Configurar um cliente de serviço para ler objetos criados pelo cliente de criptografia V1

Para ler objetos que foram gravados anteriormente usando um cliente de criptografia V1, defina `cryptoMode` como `AuthenticatedEncryption`. O trecho de código a seguir demonstra como criar um objeto de configuração com essa configuração.

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()
        .withCryptoMode(CryptoMode.AuthenticatedEncryption);
```

Configurar um cliente de serviço para obter intervalos de bytes de objetos

Para poder `get` uma faixa de bytes de um objeto S3 criptografado, habilite a nova configuração definindo `rangeGetMode`. Por padrão, essa configuração está desativada no cliente de criptografia V2. Observe que, mesmo quando ativado, um `get` de intervalo só funciona em objetos que foram criptografados usando algoritmos suportados pela configuração `cryptoMode` do cliente. Para obter mais informações, consulte CryptoRangeGetModea Referência AWS SDK for Java da API.

Se você planeja usar o Amazon S3 TransferManager para realizar downloads em várias partes de Amazon S3 objetos criptografados usando o cliente de criptografia V2, primeiro ative a `rangeGetMode` configuração no cliente de criptografia V2.

O trecho de código a seguir demonstra como configurar o cliente V2 para realizar um `get` de intervalo.

```
// 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);
```

Chave OpenPGP para o AWS SDK for Java

Todos os artefatos Maven disponíveis publicamente para o AWS SDK for Java são assinados usando o padrão OpenPGP. A chave pública necessária para verificar a assinatura de um artefato está disponível na seção a seguir.

Chave atual

A tabela a seguir mostra as principais informações do OpenPGP para as versões atuais do SDK para Java 1x e do SDK para Java 2.x.

ID da chave	0xAC107B386692DADD
Tipo	RSA
Tamanho	4096/4096
Criado	30/06/2016
Expira em	2026-09-27
ID de usuário	SDKs e ferramentas da AWS <aws-dr-tools@amazon.com>
Impressão digital da chave	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

Para copiar esta chave pública OpenPGP do SDK para Java na área de transferência, selecione o ícone “Copiar” no canto superior direito.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Hostname:
Version: Hockeypuck 2.2

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

```
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJo12ZrBQkTQxnmAAoJEKwQezhmktrdi18P/A3De83MBx8bdcWJ
Fot71Vk1TyBQFErgrtcytSU0czEHx3tGbzgQLbMlyzjirOT03usxEkOeqTVK+RU+
5uFXNZYQLwMJlHJ6S8tnfLe/ExM5WQ2KPwIUPfZs1GDDRQB2dIKSc+qYrP1O1vf4
O4iPgfLHMW2bFh3zjjxcaHCJyqc7Cau33eZFBAsRni1jOUo7MeyX0hlXfW8pd48Q
wZllQVZ/6KmDiFWA0CZ+2svJ5cL0tgPoh1OQjoz0nHpNfuDILMrZ+e7tx2VTlkGH
UGeNSydnrK8v9ztFn34KtU/k7NEWoVSyEi+5ICZL18FBwPqTwdVWXwXrqZCKiIpr
8ZdJWDz2sJfgDFNCC6rKgCQ6FrmaD9G76dYWkQ4AbZqABlUzU3q36W1K0r3iOAb5
G4tdOt4yqXHTe1x+ZUNaeW7gaCmtXAxLw0OfeJrcq/44b/SQP+qJ8sSOv76Yg2oF
BsF5DWOVUFghbTyokHAoVROyhBR4dUUisY39AqLSL8+Lp9Pr3wNuGl9GLrMD5701
piUb88B3Gwe1EiKV1gaKrvZ3mECDUiSMVO0Z5iG8E4QDpNmVbJbV1uT821ubvtOv
2Ko10Fa0uwCYGssdRGqEXNy6jz/Er8LAC3+nmGINDJQzrF+loYoSSkI2Nu7lhMuL
7iWwUPF7OhDXoVSAn4X3x6q2rGK0wsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZTsFCRNDGLYACgkQ
rBB7OGaS2t0/Ow//YIv51vHtD+kwMmIvk3zpizDHY0zW2dOezAo+C/DsSyC7wDll
Dixw34EQ1yLXH5xLR8CH1zupl3JmmEp1ucdQggoefbidxDl8Fld7tJOD1y3GGnTD
0jAl2ZC+W65Oh+wS1mD1FlaKjMGGkvJf0dA7RtU2T8dv3vt8dsxg76FMFS3+fqlC
FNOAsNTn9zWR1SqBIfkMJK83aq6s/rcEV9VrAYgDgqex58fygB5EuTf842/IF7WZ
Q9gd6fupB0mMZP5YWd2uj/vsBTYakG+mgQwDxZuKPeEzAqnqqS7biSQOUO6Wozlq
Yy4fSczE9GkBAvg0pGmbko+zHvpnjvX/h1CUpC6odvFyOAhZp6zyhs0QWz9thfqV
lU8WlbgJ2atFDn5GUSxF/fe0Yzovlbbs6sbYXuvMG9RiEOuJ1mBbZR3aIdZ1U6Do
BHc/vjc5mWcV7JQSP7i4W/8W7X3UAuN9LdxB+IvF3Cwrgtlw2BWvA5Alco5Tnz8t
P/CIVmBjk+sLme8W4kfLK3IWEbwClOdNnErI/MHRm65A2Y5EMIhwjrOi07SU1Pxa
nPpg3OYJCdvjzdB8QE3/DBiMfOl4dISfKDVEWnfK8mZaYd/BeRm2gUAa9UrqSFCG
BlA7Lg+eLI3US0FvwWJ4j5bBJqgLu+y7crIkiUOPAQuLk3lO+5uYU/I3DuLCwZQE
EwEKAD4CGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AWIQT+uSCfLy8/RmSEHlWs
EHs4ZpLa3QUCZwAXCAUJEWvKgwAKCRCsEHs4ZpLa3ZdTEACMBLg2q9zk8ZH02nDz
Sg5zc8Wlqq8WdxU0Pj8qx4UOrrMca7wyiUvrgoxPW5lh1RVNUeMkDRfu9pSXcOVI
V9LvmYE/WnwKROubgGbsC4T7M/LqV0/AulXil4d7IXcO6l4toa8LTNWtD5bODgrN
gvay1AzCU8kq1Qw1cKZ2gAfvA3Ba7PWyLeUN4HTlGrXcw73G+0CofY1L8wqWxHCJ
29XqQzeTEc6MDEeIlNlVdUcy8Qr5uwkEsl34H9AxS5F1opJ4TqvXiDZsrSRRv57R
XYmRZDWeYT+9PZaMsHXza5qgej7BfATxhYfICsNaY6MK3x6b+nDSKkoZgO+i09zh
1YjpahhQe6G336v/3mRj0dKGCRQ6znQ9ghUaB5z9zfvgH5AOEkTe3l8MqM+j5A6P
VjSBBJAHKejxr7+wKJKIA6P+DqpsYAunzftwUzrLVqb+BZQ+DcTmVrE7OPcMYJD5
QglX/Le+WmWZHI154NXgpWWUOUgZUbUge4DKrT+zCJ9iecPLKTW7OcULyXO+rjb8
8BGrD5GPlHB3dOUXXTlMKCqg3qy1Bu2KnZTQiaEEdZgSIGQbrW0JTMmmXJkKjokd
JMA4vYeg5en51G9nRQjScPngx77IxvByNyFWTJdG1ENpJpsK9TtmENcpyUJtJZTJ
```

```
ZSOIRVPP5RzR5vInuXWq6VV0BMLBlAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMB
AAIeAQIXgBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEoiBQkPj/2dAAoJEKwQ
ezhmktrdx1YP/0vvym3jgX/pwnR7K1rafZMb1iKQBr0ISG8cdbaf4pqX5vuUZnyj
w9Cl/oONn7jJjnQxOIIzuBoxne2WN28ftM2w0nVXm85mAmz2fwQz/fdKDyonXcOh
pfD2iMqn7gESjhEgRE7wMDYMDuLdqHI7OKWGVfgrh7xEmKapLh45h7cnumo2VjL9
uDYY1aOBHz993T7oE41y43rhk+6kKbGFd2uuo7h5j1ZF8Lj6sYfcEzXOU1OhRlD0
nyBjDy9MYWu0YNouc70WgMceGx6hjvCAM/5fxP7SZFecZ7ePeB0GpvVA24hSNENE
0r3tUekuOf1I0FunMnMnbh7ZO9rPYqWvWDNIpU3S4CjFhY82L+IeKnmLy8N6ASRk
HsPiNCOHSK8C/0ynrd9xLhX8Jsk/TGiQYaleoHhWkNLlZsL86QHL8SKEqkqZCQf5
AEqghDP6NEGS7lnOenA7JjIrA9KLlT7fnNWZOwFi5X+o/CymE2ytEMS0Yf3nmY4U
n9x56Wgn6J2zqB5nqOXf6NxGdAIgOBm098YEnKCIFzk+yhoDlprVpHcnd2b5f6Oq
uh8KYOEbKgpMJ3zZuWSL5kwGF1nNoYiAkonMaz9H3pOQnOMVYCUeUTDRsiO/prrd
UhNlry4TAsBMpeXnFhdLVM3vFQZVpByadGOJNmnaN/Wavw2a00UGBFa4wsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJhMqGaBQkLnlUVAAoJEKwQ
ezhmktrd2sQP/3YHM+U+BbOy1nSEAfykZ71+uCM2hkHMLdxQYWB/rBWkmg/pbu+d
r4t45RsTASrNjRcZOntlPMQRIq973ymHfpmeS+noFwvTGH7zDv1BRBR9wPrd1XUz
iSuEUHGi/fqxUVXQ5mbonzfThX8tuXeuiQmeToqoB00FYlZm6xsNnEHcjVl66mC4
IPoJLWnZJs4rOCeoRf5XvDTgX6xt5/kLYRZf79qaWGFvaZpsc1CH+rQJUdVa/D4T
7pI7hX6zy0S91z4iuC5HZUiOTF+y5auEZHGTdTWNS1kvOvfcCTi0XK/GkGL82SZu
7X2VGnpCeUnFyViRGlk+KaDG1sVyDY+lcBPg6ilr45M6MQV0iHS5OF04QNXSKt5+
UnzJH7lldgNsR6ibRMyNV3k5v3fyUcSBvIYyLORTTBiVEjQDSbk1QNqbrQlX9CWz
+EJWn16BFTmMFvxBSWPm640GncHP5J3/0MbMw3Cm90x7k8UfNANIemcrJrSxIDwm
g9cVAg3a+D+wxjrVe8jGg0ejvECpm+0yswigj5x6Lqj09A4UgdjEauN+/pn0nhBo
Gv7DzMXtM/LoDtgp6wn93qZVN2TsuHnkEk4UyntB6eWJbBdXHWUr47exiWh0dvQN
tpwCWPT6I7ZTPtA5K/zx+q9m6797BLgAkTYc6gloQL3vs1Z1S3m/hZNawsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJgmrz2BQkLBnBxAAoJEKwQ
ezhmktrd36oP/2rB2EkwSOCKC4m0heWSfDWi6OBKoEbbDtFtc6/HwqBW8SPsiKlq
zV0e3qBY/LVju04+ktJEK+EGXLnC3iC36MegrQ8zt391kEx/Zv9LIuVOCX90QIAX
dL8MVUkkjRLCFFH8pTgRy1cJYWk1X4dYdXWYc29fCwNVartNdNBhsb2ht3VJeKDE
kUivBHmkjuISDPEnI1coY7Lj0ZtY5cHdRF2eZpB0RkTBpsIt18rCYyHkERZrhmvb
j3rOyPyvOa+1/dQS8/hv5pEmbKx8cy8RdJkmbUHYatPBsjHkJSWr7O7G9VFW4GoN
9CRAI4KkbDSEDjCL5dv2pq0Sew1MkLuWJGULAMgiIUlWcOs5SZZGFSksNQrtSFV9
Z/wGocecMGkGQNXQ06JV/Fry/TvyphBlmylEqL+NLqEcEjnlz90IVu+ZA+M09J96
UlHO7V5GvBgM+QK/q/dJeMHPWrNlo1gA6Nwl/HBdM0DqzdZ2jEPvsQSABvZrPMty
+BAqEar4wqY1AH4X5ccEjO7nJQoBQSDRSki1fkBsc1nx44N/m0kHdIa0Z/Y+Mw4v
WiZhREkOospG1I4lBa3CNTVAhSs9msGsYfkqvFJGHL7sZY8XSv82GBBvA0nUNrsJ
bLBwo2FaQG9eoatRAGkqp4b/OtNtBuGeiQoNwFGbfUZTAaStj5/zZj0swsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJe+9bwBQkJZ4plAAoJEKwQ
ezhmktrd+ScP/RoaUKriVVAgLHOGs+/mnfKtnfTlClzi5dsdI9/6H0vLpmSWK/Cl
2cT6gary45VMgAeVK+H1lQXafYj+FY++I5kYoe2GrSvIXhpjaFAJyNf/dKleTsqR
Tm371i8b3FDYs5kvy2CnTbmHB8MsOGxck8/YHd1x+g8WpO2IgF89yYCSF3CAdxC3
6bHbs6Z3C3lcM/3SoWF+Yie2P8XeBMPCGp/BcjQzUcHF6G06TwDDYhixucUi6vEY
EH5JtOwVVQ7bubT8OFeOoJwVxlzYz4UoqxjKDWymarTzu03AUIT0PXPece94bJAK
mSh68ItQe3H8tSPMubERWz2tEV3lVkChDGXcC7BYQmxHseolxz/qzCtJ0iX9BvZR
```

```
dniZNeNJ/Cu8M2pDp47zdNFXzf/Q/sQ9pQlws22G2g119rWDneBku9n1vTP80/er
SB+VLTBjDiArlCY5y9+BG8wbscExJySoQxkB9j/nlMzPY5rgk0SyxsNj9GbqH+hr
EjS3/uacNwSLxGcOT2E9Teot5pfTEO6fQVq+35QhfAlP8c8jze01W/+u+wXu1Ui9
azRSzYtCHanGyyet6U1mlBpAkqkZzH6t3CA5czc9i6FbzjvFVZnbRUZIRzfISYew
lF5WqgTn2iYVdxagPRvLF5kjd696brGW9d5HwirCVGaK04VsXWlAblB9wsF9BBMB
CgAnBQJXdYAFAhsDBQkHhh+ABQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEKwQ
ezhmktrdWigP/3QWl7aO81BUWyby4HEhN4SdAoWGY/FLqO4mCtuplcnMgRUCSiL9
l2BSCTMCtUcdSWtYwOgSChN2mMsdi1U2FNR5HvNunYR/pFdqjfQurf1ZmKVeG5/4
uuKaOxMw9e8pK5uYAfs+O7gr8gu/f6/Drp7NZk3/yVKpf4WCY9oX9TA1q9O/11nN
cwS45U/d7YP+N1YM9cBXa1DnDcdfm0BlykzouAF0qd1Lwi/tmLENvybD3+2c2WsE
rlFZGSa5ZafO0tTIWXh5k6wh5FdRRycrnSyRK3B9N9+yaXfMQ0XpOypa8dqQEnCi
IsngDCJPxtTrhMWKhBFRUMzK/WZTDboTQSQDK+YVRrE4K8MtoZSKwZLV2r9O3TpX
kpbKsPVYmexerfdMeZfjZMF1bC7BmEs7jciH6JjbqAoAPnHzN0481aeNarINSViX
PQWr2mp9qShei2/RavLtx2ZNrvmGW72ZKpF8E3WWUDpBJqFVeGNRvOm3aZj8o/Hl
ewtNjcT4ouJfqlfKiULv+g7ANEMDLQTFDTg5twRdvmZlB7oTBsavf+LwxPIXhH32
IR7TX7VeicMMxmZnmZK2ANT/QBi3laf+ojVHvB+f6D74eLNq0Zqjfi/3UFNYsYjg
E+YgCqEUBpHbl61nOHwGOSsQwfap2uKKlzukD/KxH5SPBC3DYGBI+KCbzsFNBFd1
gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+hV6XulGA
HAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go7xHIxgFj
C046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FKVYR/j9ue
nEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQlQ1Kou+3
dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjypUwgp0MT
o25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGwsMDyHNqyJ
eYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6QxaZje9YSZU
ijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KDOSn5CbmX
pAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVgroUVtprs
mHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs/Hd981Fd
VghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQABwsF8BBgB
CgAmAhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZm4FCRNDGegACgkQrBB7
OGaS2t3y5g/7BFXp/fdanzuQPToJTPen7AVwhLloKaiYhG3GjdXfMPLvu6UtaaGm
qynLolUNNooobptFqc1G9BKoAghQrta7CsDHtsQF2xyc3Mfu0gmpL/7X5a7sFIeJ
j08UjfweHx4DSG4LEZgNaAoWFjZltp4+8cqijkAHXt+r+1ayQG4VVHOWyXXqmSH4
9HqtbPcPyRzxdoVLeshZC9jmhHhhKqw/LwGyipWSOUKQDjWarBwdyhNmWCaLvxH1
ndMp4tq8DPGC3G4T9tYAbANrn7nKfZgHebMSzMw9kSp0L6QvwwTDjJyIWz85WyeH
WHeBysDaBOit3XDlehUew27y7N6a9hQSYjnXuwvre5mjDIOqJon/31R6ui2Z1y9P
a+bC11hbLXXh9tLCXRuoOt6thh9Cq5X1a76PPpEv30o3bpsb6l2hbrut1OKezwvK
l7txito/jfMiWfsZHA9O4SoM+8GnmVingHtZ805n1T4RddJvT/vaqplfI6zf7jmf
a69lALP420riFOQcwntNUM5tVmFUZsnFp2YRd4Ls7MiXVjtABahlSbb94l5WSVc0
jrOLDf94edvzk4R8i2Ob8CfVZNqEsTR6bHz8dT7Q+xQzEdjUujyyZY1UUl157Qeb
OsHjhCtuZYCI04X9hZ37nKnZXSxRlRDCnt5BEiyFu2WD1RscUe6PcVDCwXwEGAEK
ACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCaNdlPQUJE0MYuAAKCRCsEHs4
ZpLa3XCpD/42DrcveE+q2ulrAIYPDlUlHiwIMejqBDRm6zmr1KSAeb4E6/MFcP4s
rXSSscMlrqG6NVynjNCXjD2YzWii68EwoXLJkgoD3r2ifzkV62EX2MIEeNZAVwuy
KNxorzmy6bhuWltRYNK/hITs2AG5orOk9ADEJ8PixKymrWlhesPaWX6Yhp9/tWaC
```

```
RHOSRiLbRVaJ+7sqT88urLmkV9Hqx949Zxv4+cgBVUGL6WXKsfWhHjbDMNJnozWB
SZaIJznLApOM8z+1DNrqUYyfR8SkF4IOVmg6HDzoyuseJJ8JvMAlkvT6F9VBq/iE
yeDYdEEQxwHwozKrEx5Ybxl5mntbqwCXy6kHSx2+/3RZWpZQ8K29YP9QEk0KeGF8
9Vap3jjNrX4u3cuRNQpeblQc4uFn3Nzaj+cVV4YzcRw94NifecXpujSvk8XU2ytJ
/JgMBxPIBKglN4eEMet9b4FRB5XeBdPAm19/LXyb4IlIipGNXlgNz/HCuBzidzHT
QQdqfA9rZVx1hwFr7AJCVqWaXVsx1oEAhKqpTtsLMyj594DvnRuwKw5Vse+1eydW
MIHYdbxmJccsTGIt/hsOpc8zfm+QYk5752jshhOKEBy+Ey3QZI1WbO547NOb2Hwr
Pgt7fw2NCKMPElSu98zmneFPhqNHf7L5urBe5gADj81E8lm6t/oVxcLBfAQYAQoA
JgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJnABcLBQkRa8qFAAoJEKwQezhm
ktrde3MP/13CLWp99XvRROrzD/bWOfWjAenT2PE/tYdOY9YcTQFbnIUhaVUDWAo3
pibR3D4u9LlY4olpGfJ7BTIHFa9myfpaVvmrNjueYI4omli24JQ/CKqNdY8Qzxz+
/QyiNK7Aw5cEBWIu84WGB1SsefWWT3rZe9YBb77gNcWHZ15pXTXrcgUxGY4808MC
I9YFWq8EA0iHawtFnmB3UFfClWt37Hy3PKvr1is3uG60+ULI8RQz3/+ZwSG8U+xt
b+I7H9+gITc1eFCb+tIwp5xWflyxcFXYk6UzOL7y3Fg2tIEuSNtIHUC9NDVobf6c
I0KAzZcMvKiPQiuBnVOjgDLmCZM5H6axj9x+gi4oVh6ea3HLqMzyjm5JkeCGgKWv
H0gD3yGEZDvcbavkQOle5T+4JefndKzCPrluX0iyx+oQiiOL8WieSSkSB6BsZcUN
SeuGJwM79Y7Oqld/YVrQNBZj5Vz+m3nZ+0EWDDMI0hRgMpSEIc+dnTC0u103Z+Rc
c2IJq8INmU653sUcfCZE12ParW4rF7ib6kViYrABT8f4e2TP0aOyP5kp51ied9qL
azaBA6tt/C9X1V2EJZK4srXtmcZO2Im45RAiVXyfpBAmmiF3eZWCbKe7qBC4rDRh
LZG4RQW/S86Da0BID7gQz9IFSkaG504MsDhvnA7iAqaHUHUepCsiwsF8BBgBCgAm
AhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiQFCQ+P/Z8ACgkQrBB7OGaS
2t3AwA/9GkXKUgvjKGCxwE4SdDt7c2jw6to2TTP9iFJ3Xbk3+5BURT3gkZCuu9D7
gt+97aVo/B4EM7Xz8DQKyY7Ic9VAwDRra/Hwi1V0hw1zyIWQ/gAnX3baU6qLRWHR
vVR5meV8r35C+rg9DaWFYmvS7PIv9LfxESwBPUjbmx8k4/5EJpHUwf12bzkTnot5
7q5lHxKQa6IvqQak+Hp9ZM2KPdsgKO2HWJJIIvYcI5byW9zBKV0O7YR8gtRAJKp9
IbtsXx0WT6cqHOFVc5SSzdcaMt0gLFl7BTnJyvKK2l9GABGBmzYDjeCyF2J+Ippf
oqxqfTe6EoOsuEMc2PbLTs9SsWjyCC2VGlX8+uUH9SoKwL0VQ6LFsP6fhkVKqi/a
rB6UuPR/iZnrKIuxMNQ4U+t2Q6UdMlmXsAXTNdkwzoK9oJRokIrH0ZV1KtH4sjjA
tCic+tOddq+GQLiKe2WpJfxlA0uESCB0TxjAwQmfn1H+dUhPeLlbNimHlH0/hXPd
ifuNGozzADIRseQDyzjl8xGL1qRZLD3cfmda6RyZ+S3dQRuaRrcFCDccpY/pO+F8
jbx64zyqqNs+KV+SkQGOcKFhWTZGCfQ/zMDtDmQKjb3eTAkv1zdEOMw9zEjjmS0q
8FNl+2wO3VnvXwvBbtDdVCIaIq+jVcsy5XtnnV+bJ19Q9yue/XvCwWUEGAEKAA8C
GwwFAmEyoZoFCQueVRUACgkQrBB7OGaS2t1uHBAAhOYVvrtchRmzCvdNER1DtkIs
bgQPJ9OxbyfvmvoD06qxH7PrycLZKbt7yYpAUU/CMc86GwaEe0I5Nm1CTs6NvDIv
g3e7EPIS859tyQflbM56NlwbsopCuoCJYknuroIf/M6dW6vJKNXLMmnL/AtalUBw
X+5pblmGUUJep49oTOxQEnvnuqyvaGjXgFXix5PVFJD2ed5NnQeFpvfCpc/ioNOj
z7ORO82j1ht5nWqPraXX5AYhQFM/kwR1cK4LV7gVDd/q+dfGYHzpxQ/HtyX/Lasi
N6I52QqA95SM1ZZLPFLaNh6EvnB7uC9pLCYS8nvilX7/cez5PFff1e1gXCOT0jv3
mJ2exLmXV0BbfKgjccFCxhrdRLtukfiDfJkySy1zdscnpfng8wJ3xKRv43cUTz7M
Z24OYNMqK26aJZVXEQUYjCwsBylY/F5wjYAwgwZ8yF5RFix28P/K8JsIHb3QrAJK
sNWQAb03ZWis3N3spR5M9Mw3VuDZ3WUXq7mxB5M3kpVoZ3vETU5cwTbADYNPf4Sw
BDK2uIVtxabezxSBtz0FcyYoF+OW8q7r4WvoyC9/+3GfnozZLJcEIVDk4W2pMW4A
UhG/6drKTm3HkSDWIDu7d1sHWMffLEYfUHtN5DKkDkGoPfHvZvu9teR5yLfUrPTf
ktihPn/JMrmwa9pwi8LCwWUEGAEKAA8CGwwFAmCavPcFCQsGcHIACgkQrBB7OGaS
```

```
2t0uaA//UWRaRiHEAKeRqBG/T2ak+XZJNu7QHfNgoUEAub9Zru8oPPXx2AJLcHEN
KWmeFlLxADdWOZs4Bm9oOew3VQnR/dBqjnXfob9Rc+eYUjA3rXazM/QrqcU8Syi3
MjNGUmjdL5aQF+IppAMgOBLG1TEnM7C5/PvrGJuYpGEnkKEwMK/GYhqg2V60pHEV
Pvs66mefJpCzbZSy56qtknSt6yBNWc14XgDX6VTn2kW4CV/3vVJUuvjvYs9SPyY8
mKEXa6QvUd3PcXv6RiWk4lGYuT1+jh2VkcFQ+JnUwv9TbKFB9b5jq1bvW9+LMDEl
YXux7pBP5RPk+OLpyiExIRFWhi3x7aMWOzQ+I9yuNTeYkTHiEAQRUhs/1Fh4oLgI
v9QZgC0mRSN3zm8plQdivs1ZlAosAqqkA9BQwqsgosQe7P92irYIJqay0si9wGCD
wSMsmeXdIF6wW3/UMJZl66aarPeiZApGX0QdTZwjMh/QK/8gTKyeZulKmNkNfwWq
O17OirWqLKssVHtg3VUM8EIdh+oNqDDXSeWtYUmpPpWp+yWZ0x1MFFZhUQHQZTGu
TIj4A92LQzbrfj/jXRvWm2SrJMivUoiDUn+qxKIpVwFlI5gVb+uyTFhw89PCkphr
JwRi052RLoU9yd6Ek46UH4XfZZWrZuzY+zzB7oqGONphLgi/h3DCwWUEGAEKAA8C
GwwFAl771b8FCQlniTUACgkQrBB7OGaS2t2/MxAAjoEGPdzavhsOlXdPCRd1D5QJ
r8T/NSEV2z1cp8ZvdrkjNF09TBP4qsBnKJiuvY1Iw7OGX9W2okvXxgJizE45v9MH
WEMz4hmIjmAfRwcqENgpOc1IY/T0/+kkCW8dB6d30J1kT0n2PCRzN9L5vPqZXGTG
mLvd9MOjH1256w4uxLb+e1HMDTCqEN1ppq9G+EAR/29q8JZWs1marbZZWxSWcg/E
1YYbNafzklgjq4CLh/j8AEWSvLr39zRy9uvQ/yqAKZ4K4aZfh/SPupGDvsD6ZK54
EPHxErQ7aiXTbUHtvwhxWLOP6WmxFA3Shr6L6YUb6jq+0PVliFC517g3mxFHJtwt
yXGNIKhmzmr0l9OlsHafulJ/9QPfK3Ce32SkPhW/11MYA8HzduMv5Arp7cBczXSP
EUTmNIVKv3gTjSQrzRhwhHmMuqyDZ/rXQQ1jl2sxIDjO4MUMvVjYKF+OCNm42gVs
8ca3/wN9ZNU6hyFWeKQDuCAqPPbT5GO/DKseFEwB+07wwyH1RXbyl0v4fneg605X
S7lqhNtw2p1hDL0HYHDiV+aPZ+LoOmX6+dmnqE6bQJaIlVb922KWmliO7F3DkqP7
0jFlhoE1gfiXWkxP4Gy8wOobNfEMgvz02djkGQy+oQqeNdIcZFZgzPTGKB/nVgpt
9CcRDWjPltFCd2e1FBbCwWUEGAEKAA8FAld1gAUCGwwFCQeGH4AACgkQrBB7OGaS
2t1PIQ//Qc5VYfBCxpaMysaPQ44wXPEZSjxIGZhhMGzb1UzzAEYOw+RgKN5nNTXq
L2KoOkOrGnKqZOKByMdXwIPH/rGwwEsbbIpopnibf5ic5B/+xCTIK+qLIwX2ZLuk
NhbL6Y+E+7DxMMh+KqBWHONKkgwVY+rFWOfoops839ABKvc9/Ry4/qqkcb40AzpD
l1iQJ5vK/DMuaDWxWeKXqJLIl3WMGPcPfheuBZL1u7LEEHYKMgzvpbF81WIn3MBo
8jvxf2/o+kMafSSDqgvOu6yu8GOhmScpCbRJn7jV/HrG+tM+zy48TN6/MkGWSR7q
TD34pqBjyatVfVl6dGD6xj/i/Emt5hZB6qXruCDH7AWMoNx+FkDubs4sc4PKysZU
Itya6KdQFo2UeYsNwZhdn6QwKhd85um4JUHJCY0mARvjsQgWXH/5MR40cow77bbE
vVq0XNd+QRVlyT42CEtnIUOFLeDVuZrum5Tuvvna6ImMDoi/z6QcNeL79XsY2m6I
QVRiHr1BDb/8JLkfnWiwL8GRv169Kf8unx0y5u1YBpcMYkyDD2+pnnk3TY0rR+8X
8goecaS8fbyu/Q48K85ZMD8wKW/bzLQ+tK9y8xed24u2QERftMhIw9b6f45Nrrf/
PhgV8RnuwUusSbdDe8kw3eYTmLdzD4kZc9K7SdO2CqT+hm//9JI=
=uGHC
-----END PGP PUBLIC KEY BLOCK-----
```

Chaves anteriores

> **Important**
>
> Novas chaves são criadas antes que as anteriores expirem. Como resultado, a qualquer momento, mais de uma chave pode ser válida. As chaves são usadas para assinar artefatos a partir do dia em que são criadas, portanto use a chave emitida mais recentemente quando a validade das chaves se sobrepõe.

Data de expiração: 2025-10-04

ID da chave	0xAC107B386692DADD
Tipo	RSA
Tamanho	4096/4096
Criado	30/06/2016
Data de expiração	2025-10-04
ID de usuário	SDKs e ferramentas da AWS <aws-dr-tools@amazon.com>
Impressão digital da chave	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

Para copiar esta chave pública OpenPGP do SDK para Java na área de transferência, selecione o ícone “Copiar” no canto superior direito.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Hostname:
Version: Hockeypuck 2.2

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

```
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJnABcIBQkRa8qDAAoJEKwQezhmktrdl1MQAIwEuDar3OTxkfTa
cPNKDnNzxaWqrxZ3FTQ+PyrHhQ6usxxrvDKJS+uCjE9bmWHVFU1R4yQNF+72lJdw
5UhX0u+ZgT9afApE65uAZuwLhPsz8upXT8C6VeKXh3shdw7qXi2hrwtM1a0Pls4O
Cs2C9rLUDMJTySrVDDVwpnaAB+8DcFrs9bIt5Q3gdOUatdzDvcb7QKh9jUvzCpbE
cInb1epDN5MRzowMR4iU2VV1RzLxCvm7CQSyXfgf0DFLkXWiknhOq9eINmytJFG/
ntFdiZFkNZ5hP709loywdfNrmqB6PsF8BPGFh8gKw1pjowrfHpv6cNIqShmA76LT
3OHViOlqGFB7obffq//eZGPR0oYJFDrOdD2CFRoHnP3N++AfkA4SRN7eXwyoz6Pk
Do9WNIEEkAcp6PGvv7AokogDo/4OqmxgC6fN+3BTOstWpv4FlD4NxOZWsTs49wxg
kPlCCVf8t75aZZkcjXng1eClZZQ5SBlRtSB7gMqtP7MIn2J5w8spNbs5xQvJc76u
NvzwEasPkY+UcHd05RddOUwoKqDerLUG7YqdlNCJoQR1mBIgZButbQlMyaZcmQqO
iR0kwDi9h6Dl6fnUb2dFCNJw+eDHvsjG8HI3IVZMl0bUQ2kmmwr1O2YQ1ynJQm0l
lMllI4hFU8/lHNHm8ie5darpVXQEwsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiIFCQ+P/Z0ACgkQ
rBB7OGaS2t3HVg//S+/KbeOBf+nCdHsrWtp9kxvWIpAGvQhIbxx1tp/impfm+5Rm
fKPD0KX+g42fuMmOdDE4gjO4GjGd7ZY3bx+0zbDSdVebzmYCbPZ/BDP990oPKidd
w6Gl8PaIyqfuARKOESBETvAwNgwO4t2ocjs4pYZV+CuHvESYpqkuHjmHtye6ajZW
Mv24NhjVo4EfP33dPugTjXLjeuGT7qQpsYV3a66juHmPVkXwuPqxh9wTNc5TU6FG
UPSfIGMPL0xha7Rg2i5zvRaAxx4bHqGO8IAz/l/E/tJkV5xnt494HQam9UDbiFI0
Q0TSve1R6S45/UjQW6cycyduHtk72s9ipa9YM0ilTdLgKMWFjzYv4h4qeYvLw3oB
JGQew+I0I4dIrwL/TKet33EuFfwmyT9MaJBhqV6geFaQ0uVmwvzpAcvxIoSqSpkJ
B/kASqCEM/o0QZLuWc56cDsmMisD0ouVPt+c1Zk7AWLlf6j8LKYTbK0QxLRh/eeZ
jhSf3HnpaCfonbOoHmeo5d/o3EZ0AiA4GbT3xgScoIgXOT7KGgOWmtWkdyd3Zvl/
o6q6Hwpg4RsqCkwnfNm5ZIvmTAYXWc2hiICSicxrP0fek5Cc4xVgJR5RMNGyI7+m
ut1SE2WvLhMCwEyl5ecWF0tUze8VBlWkHJp0Y4k2ado39Zq/DZrTRQYEVrjCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmEyoZoFCQueVRUACgkQ
rBB7OGaS2t3axA//dgcz5T4Fs7LWdIQB/KRnvX64IzaGQcwt3FBhYH+sFaSaD+lu
752vi3jlGxMBKs2NFxk6e2U8xBEir3vfKYd+mZ5L6egXC9MYfvMO/UFEFH3A+t3V
dTOJK4RQcaL9+rFRVdDmZuifN9OFfy25d66JCZ5OiqgHTQViVmbrGw2cQdyNWXrq
YLgg+gktadkmzis4J6hF/le8NOBfrG3n+QthFl/v2ppYYW9pmmxzUIf6tAlR1Vr8
PhPukjuFfrPLRL3XPiK4LkdlSI5MX7Llq4RkcZN1NY1LWS8699wJOLRcr8aQYvzZ
Jm7tfZUaekJ5ScXJWJEaWT4poMbWxXINj6VwE+DqKWvjkzoxBXSIdLk4XThA1dIq
3n5SfMkfuWV2A2xHqJtEzI1XeTm/d/JRxIG8hjIs5FNMGJUSNANJuTVA2putCVf0
JbP4QlafXoEVOYwW/EFJY+brjQadwc/knf/QxszDcKb3THuTxR80A0h6ZysmtLEg
PCaD1xUCDdr4P7DGOtV7yMaDR6O8QKmb7TKzCKCPnHouqPT0DhSB2MRq437+mfSe
EGga/sPMxe0z8ugO2CnrCf3eplU3ZOy4eeQSThTKe0Hp5YlsF1cdZSvjt7GJaHR2
```

```
9A22nAJY9PojtlM+0Dkr/PH6r2brv3sEuACRNhzqCWhAve+zVnVLeb+Fk1rCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmCavPYFCQsGcHEACgkQ
rBB7OGaS2t3fqg//asHYSTBI4IoLibSF5ZJ8NaLo4EqgRtsO0W1zr8fCoFbxI+yI
qWrNXR7eoFj8tWO7Tj6S0kQr4QZcucLeILfox6CtDzO3f3WQTH9m/0si5U4Jf3RA
gBd0vwxVSSSNEsIUUfylOBHLVwlhaTVfh1h1dZhzb18LA1Vqu0100GGxvaG3dUl4
oMSRSK8EeaSO4hIM8ScjVyhjsuPRm1jlwd1EXZ5mkHRGRMGmwi3XysJjIeQRFmuG
a9uPes7I/K85r7X91BLz+G/mkSZsrHxzLxF0mSZtQdhq08GyMeQlJavs7sb1UVbg
ag30JEAjgqRsNIQOMIvl2/amrRJ7DUyQu5YkZQsAyCIhSVZw6zlJlkYVKSw1Cu1I
VX1n/Aahx5wwaQZA1dDTolX8WvL9O/KmEGWbKUSov40uoRwSOeXP3QhW75kD4zT0
n3pSUc7tXka8GAz5Ar+r90l4wc9as2WjWADo3CX8cF0zQOrN1naMQ++xBIAG9ms8
y3L4ECoRqvjCpjUAfhflxwSM7uclCgFBINFKSLV+QGxzWfHjg3+bSQd0hrRn9j4z
Di9aJmFESQ6iykbUjiUFrcI1NUCFKz2awaxh+Sq8UkYcvuxljxdK/zYYEG8DSdQ2
uwlssHCjYVpAb16hq1EAaSqnhv86020G4Z6JCg3AUZt9RlMBpK2Pn/NmPSzCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAl771vAFCQlnimUACgkQ
rBB7OGaS2t35Jw/9GhpQquJVUCAsc4az7+ad8q2d9OUKXOLl2x0j3/ofS8umZJYr
8KXZxPqBqvLjlUyAB5Ur4fWVBdp9iP4Vj74jmRih7YatK8heGmNoUAnI1/90qV5O
ypFObfvWLxvcUNizmS/LYKdNuYcHwyw4bFyTz9gd3XH6Dxak7YiAXz3JgJIXcIB3
ELfpsduzpncLeVwz/dKhYX5iJ7Y/xd4Ew8Ian8FyNDNRwcXobTpPAMNiGLG5xSLq
8RgQfkm07BVVDtu5tPw4V46gnBXGXNjPhSirGMoNbKZqtPO7TcBQhPQ9c95x73hs
kAqZKHrwi1B7cfy1I8y5sRFbPa0RXeVWQKEMZdwLsFhCbEex6iXHP+rMK0nSJf0G
9lF2eJk140n8K7wzakOnjvN00VfN/9D+xD2lCXCzbYbaDXX2tYOd4GS72fW9M/zT
96tIH5UtMGMOICuUJjnL34EbzBuxwTEnJKhDGQH2P+eUzM9jmuCTRLLGw2P0Zuof
6GsSNLf+5pw3BIvEZw5PYT1N6i3ml9MQ7p9BWr7flCF8CU/xzyPN7TVb/677Be7V
SL1rNFLNi0IdqcbLJ63pTWaUGkCSqRnMfq3cIDlzNz2LoVvOO8VVmdtFRkhHN8hJ
h7CUXlaqBOfaJhV3FqA9G8sXmSN3r3pusZb13kfCKsJUZorThWxdaUBuUH3CwX0E
EwEKACcFAld1gAUCGwMFCQeGH4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ
rBB7OGaS2t1aKA//dBaXto7zUFRbJvLgcSE3hJ0ChYZj8Uuo7iYK26mVycyBFQJK
Iv2XYFIJMwK1Rx1Ja1jA6BIKE3aYyx2LVTYU1Hke826dhH+kV2qN9C6t/VmYpV4b
n/i64po7EzD17ykrm5gB+z47uCvyC79/r8Ouns1mTf/JUql/hYJj2hf1MDWr07/X
Wc1zBLjlT93tg/43Vgz1wFdrUOcNx1+bQGXKTOi4AXSp3UvCL+2YsQ2/JsPf7ZzZ
awSuUVkZJrllp87S1MhZeHmTrCHkV1FHJyudLJErcH0337Jpd8xDRek7Klrx2pAS
cKIiyeAMIk/G1OuExYqEEVFQzMr9ZlMNuhNBJAMr5hVGsTgrwy2hlIrBktXav07d
OleSlsqw9ViZ7F6t90x5l+NkwXVsLsGYSzuNyIfomNuoCgA+cfM3TjzVp41qsg1J
WJc9Bavaan2pKF6Lb9Fq8u3HZk2u+YZbvZkqkXwTdZZQOkEmoVV4Y1G86bdpmPyj
8eV7C02NxPii4l+qV8qJQu/6DsA0QwMtBMUNODm3BF2+ZmUHuhMGxq9/4vDE8heE
ffYhHtNftV6JwwzGZmeZkrYA1P9AGLeVp/6iNUe8H5/oPvh4s2rRmqN+L/dQU1ix
iOAT5iAKoRQGkduXrWc4fAY5KxDB9qna4oqXO6QP8rEflI8ELcNgYEj4oJvOwU0E
V3WABQEQALzM0Cs9Zvd08xOEvbEBj59LrS9d0HVKQ61gmkNakWC+jR35VD6FXpe6
UYAcBLrEbVYfKw9P0p6MhFKAsb570JoznKGzE1rVYUZQzhD0RKje35rvkajvEcjG
AWMLTjr87pWHeD0389ER64bzORncfa/l+YP56PI+CThb2wUvTTONGJkPQUpVhH+P
256cQL/Y0Fwu4XLerpwN+YKgMQ47raRcydobPeSfMQr9fVKRyOzFEOrvNpCVDUqi
77d0gLDLjHlIlDyOX5554S8XYLb91eYOiFvnu2pTCKiiExRCSYK29mAQePKlTCCn
QxOjbmBbGS8mVIkpQ5vpvXvzpY3JIjMXaDGqWSQSYGXhECyxCR5eOtKYbCwwPIc2
rIl5gW6yXyw9pKmj5XafTP7YHTvRSr7CZ/VLkDkWl6AfQ9nPOg1mjwjpDFpmN71h
```

```
JlSKMaZkh0QGV5FW3dK+GLwxiWdqx3htbZErWyvumWQF/xBF7puKJBEXcoM5KfkJ
uZekBwcnVkfNFF2RdkM1ALq8InGzLXc7ROuEm0BXVirfju7JRtWLb3UhJWCuhRW2
muyYegSTkag5MduD1IJK37GL8WIlAL65taYgZegUoxHdSaEOefOhspxuduz8d33z
UV1WCFhi+r/+BMCQmTRbF8ao7fTC1dGd084DRP6qE/dMT4u0ZEn7ABEBAAHCwXwE
GAEKACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCZwAXCwUJEWvKhQAKCRCs
EHs4ZpLa3XtzD/9dwi1qffV70UTq8w/21jn1owHp09jxP7WHTmPWHE0BW5yFIWlV
A1gKN6Ym0dw+LvS5WOKJaRnyewUyBxWvZsn6Wlb5qzY7nmCOKJpYtuCUPwiqjXWP
EM8c/v0MojSuwMOXBAViLvOFhgdUrHn1lk962XvWAW++4DXFh2deaV0163IFMRmO
PNPDAiPWBVqvBANIh2sLRZ5gd1BXwpVrd+x8tzyr69YrN7hutPlCyPEUM9//mcEh
vFPsbW/iOx/foCE3NXhQm/rSMKecVn5csXBV2JOlMzi+8txYNrSBLkjbSB1AvTQ1
aG3+nCNCgM2XDLyoj0IrgZ1To4Ay5gmTOR+msY/cfoIuKFYenmtxy6jM8o5uSZHg
hoClrx9IA98hhGQ73G2r5EDpXuU/uCXn53Sswj65bl9IssfqEIoji/FonkkpEgeg
bGXFDUnrhicDO/WOzqpXf2Fa0DQWY+Vc/pt52ftBFgwzCNIUYDKUhCHPnZ0wtLtd
N2fkXHNiCavCDZlOud7FHHwmRNdj2q1uKxe4m+pFYmKwAU/H+Htkz9Gjsj+ZKedY
nnfai2s2gQOrbfwvV9VdhCWSuLK17ZnGTtiJuOUQIlV8n6QQJpohd3mVgmynu6gQ
uKw0YS2RuEUFv0vOg2tASA+4EM/SBUpGhudODLA4b5wO4gKmh1B1HqQrIsLBfAQY
AQoAJgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEokBQkPj/2fAAoJEKwQ
ezhmktrdwMAP/RpFylIL4yhgscBOEnQ7e3No8OraNk0z/YhSd125N/uQVEU94JGQ
rrvQ+4Lfve2laPweBDO18/A0CsmOyHPVQMA0a2vx8ItVdIcNc8iFkP4AJ1922lOq
i0Vh0b1UeZnlfK9+Qvq4PQ2lhWJr0uzyL/S38REsAT1I25sfJOP+RCaR1MH9dm85
E56Lee6uZR8SkGuiL6kGpPh6fWTNij3bICjth1iSSCL2HCOW8lvcwSldDu2EfILU
QCSqfSG7bF8dFk+nKhzhVXOUks3XGjLdICxZewU5ycryitpfRgARgZs2A43gshdi
fiKaX6Ksan03uhKDrLhDHNj2y07PUrFo8ggtlRpV/PrlB/UqCsC9FUOixbD+n4ZF
Sqov2qwelLj0f4mZ6yiLsTDUOFPrdkOlHTJZl7AF0zXZMM6CvaCUaJCKx9GVdSrR
+LI4wLQonPrTnXavhkC4intlqSX8ZQNLhEggdE8YwMEJn59R/nVIT3i5WzYph5R9
P4Vz3Yn7jRqM8wAyEbHkA8s45fMRi9akWSw93H5nWukcmfkt3UEbmka3BQg3HKWP
6TvhfI28euM8qqjbPilfkpEBjnChYVk2Rgn0P8zA7Q5kCo293kwJL9c3RDjMPcxI
45ktKvBTZftsDt1Z718LwW7Q3VQiGiKvo1XLMuV7Z51fmydfUPcrnv17wsFlBBgB
CgAPAhsMBQJhMqGaBQkLnlUVAAoJEKwQezhmktrdbhwQAITmFb67XIUZswr3TREd
Q7ZCLG4EDyfTsW8n75r6A9OqsR+z68nC2Sm7e8mKQFFPwjHPOhsGhHtCOTZtQk7O
jbwyL4N3uxDyEvOfbckH5WzOejZcG7KKQrqAiWJJ7q6CH/zOnVurySjVyzJpy/wL
WpVAcF/uaW5ZhlFCXqePaEzsUBJ757qsr2ho14BV4seT1RSQ9nneTZ0Hhab3wqXP
4qDTo8+zkTvNo9YbeZ1qj62l1+QGIUBTP5MEdXCuC1e4FQ3f6vnXxmB86cUPx7cl
/y2rIjeiOdkKgPeUjNWWSzxS2jYehL5we7gvaSwmEvJ74pV+/3Hs+TxX39XtYFwj
k9I795idnsS5l1dAW3yoI3HBQsYa3US7bpH4g3yZMkstc3bHJ6X54PMCd8Skb+N3
FE8+zGduDmDTKitumiWVVxEFGIwsLAcpWPxecI2AMIMGfMheURYsdvD/yvCbCB29
0KwCSrDVkAG9N2VorNzd7KUeTPTMN1bg2d1lF6u5sQeTN5KVaGd7xE1OXME2wA2D
T3+EsAQytriFbcWm3s8Ugbc9BXMmKBfjlvKu6+Fr6Mgvf/txn56M2SyXBCFQ5OFt
qTFuAFIRv+nayk5tx5Eg1iA7u3dbB1jH3yxGH1B7TeQypA5BqD3x72b7vbXkeci3
1Kz035LYoT5/yTK5sGvacIvCwsFlBBgBCgAPAhsMBQJgmrz3BQkLBnByAAoJEKwQ
ezhmktrdLmgP/1FkWkYhxACnkagRv09mpPl2STbu0B3zYKFBALm/Wa7vKDz18dgC
S3BxDSlpnhZS8QA3VjmbOAZvaDnsN1UJ0f3Qao5136G/UXPnmFIwN612szP0K6nF
PEsotzIzRlJo3S+WkBfiKaQDIDgSxtUxJzOwufz76xibmKRhJ5ChMDCvxmIaoNle
tKRxFT77OupnnyaQs22UsueqrZJ0resgTVnNeF4A1+lU59pFuAlf971SVLr472LP
```

```
Uj8mPJihF2ukL1Hdz3F7+kYlpOJRmLk9fo4dlZHBUPiZ1ML/U2yhQfW+Y6tW71vf
izAxJWF7se6QT+UT5Pji6cohMSERVoYt8e2jFjs0PiPcrjU3mJEx4hAEEVIbP9RY
eKC4CL/UGYAtJkUjd85vKZUHYr7NWZQKLAKqpAPQUMKrIKLEHuz/doq2CCamstLI
vcBgg8EjLJnl3SBesFt/1DCWZeummqz3omQKRl9EHU2cIzIf0Cv/IEysnmbpSpjZ
DX8Fqjtezoq1qiyrLFR7YN1VDPBCHYfqDagw10nlrWFJqT6VqfslmdMdTBRWYVEB
0GUxrkyI+APdi0M2634/410b1ptkqyTIr1KIg1J/qsSiKVcBZSOYFW/rskxYcPPT
wpKYaycEYtOdkS6FPcnehJOOlB+F32WVq2bs2Ps8we6KhjjaYS4Iv4dwwsFlBBgB
CgAPAhsMBQJe+9W/BQkJZ4k1AAoJEKwQezhmktrdvzMQAI6BBj3c2r4bDpV3TwkX
dQ+UCa/E/zUhFds9XKfGb3a5IzRdPUwT+KrAZyiYrr2NSMOzhl/VtqJL18YCYsxO
Ob/TB1hDM+IZiI5gH0cHKhDYKTnNSGP09P/pJAlvHQend9CdZE9J9jwkczfS+bz6
mVxkxpi73fTDox9duesOLsS2/ntRzA0wqhDdaaavRvhAEf9vavCWVrNZmq22WVsU
lnIPxNWGGzWn85JYI6uAi4f4/ABFkry69/c0cvbr0P8qgCmeCuGmX4f0j7qRg77A
+mSueBDx8RK0O2ol021B7b8IcVizj+lpsRQN0oa+i+mFG+o6vtD1ZYhQude4N5sR
RybcLclxjSCoZs5q9JfTpbB2n7pSf/UD3ytwnt9kpD4Vv9dTGAPB83bjL+QK6e3A
XM10jxFE5jSFSr94E40kK80YcIR5jLqsg2f610ENY5drMSA4zuDFDL1Y2ChfjgjZ
uNoFbPHGt/8DfWTVOochVnikA7ggKjz20+RjvwyrHhRMAftO8MMh9UV28pdL+H53
oOtOV0u5aoTbcNqdYQy9B2Bw4lfmj2fi6Dpl+vnZp6hOm0CWiJVW/dtilppYjuxd
w5Kj+9IxZYaBNYH4l1pMT+BsvMDqGzXxDIL89NnY5BkMvqEKnjXSHGRWYMz0xigf
51YKbfQnEQ1oz5bRQndntRQWwsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQ
ezhmktrdTyEP/0HOVWHwQsaWjMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCje
ZzU16i9iqDpDqxpyqmTigcjHV8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF
9mS7pDYWy+mPhPuw8TDIfiqgVhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+
NAM6Q5dYkCebyvwzLmg1sVnil6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNVi
J9zAaPI78X9v6PpDGn0kg6oLzrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB
lkke6kw9+KagY8mrVX1ZenRg+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHOD
ysrGVCLcmuinUBaNlHmLDcGYXZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKM
O+22xL1atFzXfkEVZck+NghLZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7
GNpuiEFUYh69QQ2//CS5H51osC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02N
K0fvF/IKHnGkvH28rv0OPCvOWTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+O
Ta63/z4YFfEZ7sFLrEm3Q3vJMN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=bboB
-----END PGP PUBLIC KEY BLOCK-----
```

Data de expiração: 2024-10-08

ID da chave	0xAC107B386692DADD
Tipo	RSA
Tamanho	4096/4096
Criado	30/06/2016

Data de expiração	2024-10-08
ID de usuário	SDKs e ferramentas da AWS <aws-dr-tools@amazon.com>
Impressão digital da chave	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

Para copiar esta chave pública OpenPGP do SDK para Java na área de transferência, selecione o ícone “Copiar” no canto superior direito.

```
-----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-----
```

Histórico do documento

Esta página lista alterações importantes feitas no Guia do desenvolvedor do AWS SDK for Java ao longo do seu histórico.

Este guia foi publicado em: 1.º de outubro de 2025.

1.º de outubro de 2025

Adição de uma nova chave PGP que expira em 2026-09-27.

5 de outubro de 2024

Atualização da informação da chave OpenPGP atual.

4 de setembro de 2024

Adicione informações sobre endpoints baseados em contas da AWS para o DynamoDB. Consulte , the section called “Usar endpoints baseados em conta da AWS”.

21 de maio de 2024

Remova as instruções para definir a propriedade de segurança `networkaddress.cache.ttl` usando uma propriedade de sistema de linha de comando java. Consulte , Como definir o TTL da JVM.

12 de janeiro de 2024

Adicione um banner que anuncie o fim do suporte para o AWS SDK for Java v1.x.

6 de dezembro de 2023

- Forneça a chave OpenPGP atual.

14 de março de 2023

- Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM.

28 de julho de 2022

- Um alerta que estamos desativando o EC2-Classic em 15 de agosto de 2022 foi adicionado.

22 de março de 2018

- Removido o gerenciamento das sessões Tomcat no exemplo de DynamoDB, já que essa ferramenta não é mais suportada.

2 de nov de 2017

- Adicionados exemplos de criptografia para o cliente de criptografia do Amazon S3, incluindo novos tópicos: Usar a criptografia do Amazon S3 do lado do cliente, Criptografia do lado do cliente do Amazon S3 com chaves gerenciadas do AWS KMS e Criptografia do lado do cliente do Amazon S3 com chaves mestras de cliente.

14 de abril de 2017

- Feitas algumas atualizações feitas na seção Exemplos do Amazon S3 usando o AWS SDK for Java, inclusive tópicos novos: Gerenciar permissões de acesso ao Amazon S3 para buckets e objetos e Configurar um bucket do Amazon S3 como um site.

4 de abril de 2017

- Um novo tópico, Ativar métricas para o AWS SDK for Java, descreve como gerar um aplicativo e métricas de desempenho do SDK para o AWS SDK for Java.

3 de abril de 2017

- Adicionados novos exemplos do CloudWatch à seção Exemplos do CloudWatch usando o AWS SDK for Java: Obter métricas do CloudWatch, Publicar dados de métrica personalizados, Trabalhar com alarmes do CloudWatch, Usar ações de alarme no CloudWatch e Enviar eventos para o CloudWatch

27 de março de 2017

- Foram adicionados mais exemplos do Amazon EC2 à seção Exemplos do Amazon EC2 usando o AWS SDK for Java: Gerenciar instâncias do Amazon EC2, Usar endereços IP elásticos no Amazon EC2, Usar regiões e zonas de disponibilidade, Trabalhar com pares de chaves do Amazon EC2 e Trabalhar com grupos de segurança no Amazon EC2.

21 de março de 2017

- Adicionado um novo conjunto de exemplos do IAM; à seção Exemplos do IAM usando o AWS SDK for Java: Gerenciar chaves de acesso do IAM, Gerenciar usuários do IAM, Usar aliases de conta do IAM, Trabalhar com políticas do IAM e Trabalhar com certificados de servidor do IAM

13 de março de 2017

- Adicionados três novos tópicos à seção do Amazon SQS: Habilitar sondagem longa para filas de mensagens do Amazon SQS, Definir tempo limite de visibilidade no Amazon SQS e Usar fila de mensagens não entregues no Amazon SQS.

26 de janeiro de 2017

- Adicionados um novo tópico do Amazon S3, Usar TransferManager em operações do Amazon S3 e um novo tópico Práticas recomendadas para o desenvolvimento da AWS com o AWS SDK for Java na seção Usar o AWS SDK for Java.

16 de janeiro de 2017

- Adicionados um novo tópico do Amazon S3, Gerenciar acesso a buckets do Amazon S3 usando políticas de bucket, e dois novos tópicos do Amazon SQS, Trabalhar com filas de mensagens do Amazon SQS e Enviar, receber e excluir mensagens do Amazon SQS.

16 de dezembro de 2016

- Adicionados novos tópicos de exemplo para o DynamoDB: Trabalhar com tabelas no DynamoDB e Trabalhar com itens no DynamoDB.

26 de setembro de 2016

- Os tópicos na seção Avançado foram movidos para Usar o AWS SDK for Java, porque eles, na verdade, são centrais ao uso do SDK.

25 de agosto de 2016

- Um novo tópico, Criar clientes de serviço, foi adicionado a Usar o AWS SDK for Java, que demonstra como usar compiladores de cliente para simplificar a criação de clientes de AWS service (Serviço da AWS).

  A seção Exemplos de código do AWS SDK for Java foi atualizada com novos exemplos para o S3 com suporte de um repositório no GitHub que contém o exemplo de código completo.

02 de maio de 2016

- Um novo tópico, Programação assíncrona, foi adicionado à seção Usar o AWS SDK for Java, descrevendo como trabalhar com métodos de cliente assíncronos que retornam objetos `Future` ou que utilizam um `AsyncHandler`.

26 de abril de 2016

- O tópico Requisitos de certificado SSL foi removido, porque deixou de ser relevante. O suporte para certificados assinados SHA-1 foi substituído em 2015, e o site que hospedava os scripts de teste foi removido.

14 de março de 2016

- Adicionado um novo tópico à seção do Amazon SWF: Tarefas lambda, que descreve como implementar um fluxo de trabalho do Amazon SWF que chama funções Lambda como tarefas como uma alternativa a usar atividades do Amazon SWF tradicionais.

4 de março de 2016

- A seção Exemplos do Amazon SWF usando o AWS SDK for Java foi atualizada com novo conteúdo:
  - Conceitos básicos do Amazon SWF: fornece informações básicas sobre como incluir o SWF nos projetos.
  - Compilar um aplicativo do Amazon SWF simples: um novo tutorial que apresenta orientação passo a passo para desenvolvedores do Java que estejam começando no Amazon SWF.
  - Desligar operadores de atividade e fluxo de trabalho sem problemas: descreve como é possível desligar classes de operador do Amazon SWF usando classes de simultaneidade do Java.

23 de fevereiro de 2016

- A origem do Guia do desenvolvedor do AWS SDK for Java foi migrada para aws-java-developer-guide.

28 de dezembro de 2015

- O the section called “Definir o JVM TTL para pesquisas de nome DNS” foi migrado de Avançado para Usar o AWS SDK for Java, e foi reescrito para fins de clareza.

  Usar o SDK com o Apache Maven foi atualizado com informações sobre como incluir a lista de materiais (BOM) do SDK no projeto.

04 de agosto de 2015

- Requisitos do certificado SSL é um novo tópico da seção Conceitos básicos que descreve a migração da AWS para certificados assinados por SHA256 para conexões SSL, além de como corrigir o 1.6 inicial e os ambientes Java anteriores para usar esses certificados, que são obrigatórios para acessar a AWS depois de 30 de setembro de 2015.

  > **Note**
  >
  > O Java 1.7+ já é capaz de trabalhar com certificados assinados por SHA256.

14 de maio de 2014

- O material de introdução e conceitos básicos foi amplamente revisado para dar suporte à nova estrutura do guia e já inclui orientação sobre como Configurar credenciais e regiões da AWS para desenvolvimento.

A discussão de exemplos de código foi migrada para o próprio tópico na seção Documentação e recursos adicionais.

As informações sobre como visualizar o histórico de revisões do SDK foram migradas para a introdução.

9 de maio de 2014

- A estrutura geral da documentação do AWS SDK for Java foi simplificada, e os tópicos Conceitos básicos e Documentação e recursos adicionais foram atualizados.

  Novos tópicos foram adicionados:

  - Trabalhar com credenciais da AWS: aborda as diversas maneiras como você pode especificar credenciais a serem usadas com o AWS SDK for Java.
  - Usar perfis do IAM para conceder acesso a recursos da AWS no Amazon EC2: apresenta informações sobre como especificar credenciais para aplicativos em execução em instâncias do EC2 de maneira segura.

9 de setembro de 2013

- Este tópico, Histórico de documentos, acompanha alterações feitas no Guia do desenvolvedor do AWS SDK for Java. Ele deve ser um complemento do histórico de notas de release.