

Guide du développeur pour la version 1.x

AWS SDK for Java 1. x



AWS SDK for Java 1. x: Guide du développeur pour la version 1.x

Table of Contents

.....	viii
AWS SDK for Java1.x	1
Version 2 du SDK publiée	1
Documentation et ressources supplémentaires	1
Prise en charge de l'IDE Eclipse	2
Développement d'applications pour Android	2
Affichage de l'historique des révisions du kit SDK	2
Génération de la documentation de référence Java pour les versions précédentes du kit SDK	2
Démarrer	4
Configuration de base	4
Présentation	4
Possibilité de se connecter au portail d'AWSaccès	5
Configurer des fichiers de configuration partagés	5
Installation d'un environnement de développement Java	7
Moyens d'obtenir le AWS SDK for Java	8
Prérequis	8
Utiliser un outil de construction	8
Télécharger le fichier jar prédéfini	8
Construire à partir des sources	9
Création d'outils	10
Utiliser le SDK avec Apache Maven	10
Utiliser le SDK avec Gradle	14
Informations d'identification temporaires et région sous un angle	18
Configurer les informations d'identification temporaires sous	18
Actualisation des informations d'identification de l'IMDS	19
Réglez leRégion AWS	20
À l'aide du AWS SDK for Java	22
Meilleures pratiques en matière AWS de développement avec AWS SDK for Java	22
S3	22
Création de clients de service	23
Obtention d'un générateur client	24
Création de clients asynchrones	25
En utilisant DefaultClient	26
Cycle de vie des clients	26

Fournir des informations d'identification temporaires	26
Utilisation de la chaîne de fournisseur d'informations d'identification par défaut	27
Spécifiez un fournisseur d'informations d'identification ou une chaîne de fournisseurs	31
Spécifiez explicitement les informations d'identification temporaires	31
Plus d'informations	32
Région AWS Sélection	32
Vérification de la disponibilité du service dans une région	32
Choisir une région	33
Choix d'un point de terminaison spécifique	33
Déterminer automatiquement la région à partir de l'environnement	34
Gestion des exceptions	35
Pourquoi des exceptions non contrôlées ?	36
AmazonServiceException (et sous-classes)	36
AmazonClientException	37
Programmation asynchrone	37
Objets Future Java	37
Rappels asynchrones	39
Bonnes pratiques	41
Enregistrement AWS SDK for Java des appels	41
Téléchargement du fichier JAR Log4J	42
Définition du chemin de classe	42
Erreurs et avertissements propres au service	43
Journalisation récapitulative des demandes et des réponses	43
Journalisation du réseau filaire détaillée	44
Journalisation des métriques de latence	45
Configuration de client	46
Configuration de proxy	46
Configuration du transport HTTP	46
Conseils sur la taille de la mémoire tampon du socket TCP	48
Stratégies de contrôle d'accès	49
Amazon S3 Exemple	49
Amazon SQS Exemple	50
Exemple Amazon SNS	50
Définissez le TTL de la JVM pour les recherches de noms DNS	51
Comment configurer le JVM TTL	51
Activation des métriques pour AWS SDK for Java	52

Comment activer la génération de métriques du SDK Java	52
Types de métrique disponibles	54
En savoir plus	56
Exemples de code	58
AWS SDK for Java2.x	58
Amazon CloudWatchExemples	58
Obtention de métriques à partir de CloudWatch	59
Publication de données de métriques personnalisées	61
Utilisation d' CloudWatch Alarmes	62
Utilisation d'actions d'alarme dans CloudWatch	65
Envoi d'événements à CloudWatch	67
Amazon DynamoDBExemples	70
Utilisation de tables dansDynamoDB	70
Utilisation des éléments dansDynamoDB	77
Amazon EC2Exemples	84
Didacticiel : Démarrage d'une instance EC2	85
Utilisation de rôles IAM pour accorder l'accès àAWSRessources surAmazon EC2	90
Didacticiel : Amazon EC2 Spot instances	97
Didacticiel : Advanced (Avancé)Amazon EC2Gestion des demandes Spot	108
Gestion des instances Amazon EC2	126
Utilisation d'adresses IP élastiques dans Amazon EC2	131
Utiliser les régions et les zones de disponibilité	134
Utilisation de paires de clés Amazon EC2	137
Utilisation des groupes de sécurité dans Amazon EC2	139
AWS Identity and Access ManagementExemples (IAM)	143
Gestion des clés d'accès IAM	144
Gestion des utilisateurs IAM	148
Utilisation des alias de compte IAM	152
Utilisation des stratégies IAM	154
Utilisation des certificats de serveur IAM	159
AmazonLambdaExemples	162
Opérations de service	163
Amazon PinpointExemples	167
Création et suppression d'applications dansAmazon Pinpoint	167
Création de points de terminaison dansAmazon Pinpoint	169
Création de segments dansAmazon Pinpoint	171

Création de campagnes dans Amazon Pinpoint	173
Mise à jour des canaux dans Amazon Pinpoint	174
Amazon S3 Exemples	176
Création, affichage et suppression Amazon S3 Compartiments	176
Exécution d'opérations sur Amazon S3 des objets	181
Gestion d'Amazon S3 Autorisations d'accès des compartiments et objets	187
Gestion de l'accès à Amazon S3 Utilisation de stratégies de compartiment	191
A l'aide de Transfer Manager pour Amazon S3 Opérations	194
Configuration d'un Amazon S3 Bucket en tant que site web	207
Utiliser Amazon S3 Chiffrement côté client	210
Amazon SQS Exemples	217
Utilisation d'Amazon SQS Feux d'attente de messages	217
Envoi, réception et suppression Amazon SQS Messages	220
Activation de l'interrogation longue pour Amazon SQS Files d'attente de messages	223
Définition du délai de visibilité dans Amazon SQS	225
Utilisation des files d'attente de lettres mortes dans Amazon SQS	228
Amazon SWF Exemples	230
Principes de base SWF	231
Création d'une Amazon SWF application simple	233
Tâches Lambda	253
Arrêt normal des travaux d'activité et de flux de travail	258
Enregistrement de domaines	261
Affichage des domaines	262
Exemples de code inclus avec le kit SDK	262
Comment obtenir les exemples	263
Génération et exécution d'exemples à l'aide de la ligne de commande	263
Génération et exécution des exemples à l'aide de l'IDE Eclipse	264
Sécurité	266
Protection des données	266
Application d'une version minimale de TLS	268
Comment vérifier la version de TLS	268
Application d'une version minimale de TLS	268
Gestion de l'identité et des accès	269
Public ciblé	269
Authentification par des identités	270
Gestion des accès à l'aide de politiques	274

Comment Services AWS travailler avec IAM	276
Résolution des problèmes AWS d'identité et d'accès	277
Validation de la conformité	279
Résilience	280
Sécurité de l'infrastructure	281
Migration du client de chiffrement S3	282
Prérequis	282
Présentation de la migration	282
Mettre à jour les clients existants pour lire les nouveaux formats	282
Migrer les clients de chiffrement et de déchiffrement vers la version V2	284
Exemples supplémentaires	286
Clé OpenPGP	288
Clé actuelle	288
Historique du document	290

Nous avons [annoncé](#) la sortie prochaine end-of-support de AWS SDK for Java (v1). Nous vous recommandons de migrer vers la [AWS SDK for Java version v2](#). Pour les dates, les détails supplémentaires et les informations sur la façon de migrer, reportez-vous à l'annonce associée.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.

Guide du développeur -AWS SDK for Java 1.x

[AWS SDK for Java](#) fournit une API Java pour les AWS services. Le kit SDK vous permet de développer facilement des applications Java qui fonctionnent avec Amazon S3, Amazon EC2, DynamoDB et plus encore. Nous ajoutons régulièrement la prise en charge de nouveaux services au kit AWS SDK for Java. Pour obtenir la liste des services pris en charge et de leurs versions d'API incluses avec chaque version du kit SDK, consultez les [notes de mise à jour](#) de la version que vous utilisez.

Version 2 du SDK publiée

Jetez un œil à la nouvelle version AWS SDK for Java 2.x à l'[adresse https://github.com/aws/aws-sdk-java-v2/](https://github.com/aws/aws-sdk-java-v2/). Il inclut des fonctionnalités très attendues, telles qu'un moyen d'intégrer une implémentation HTTP. Pour commencer, consultez le [Guide du développeur AWS SDK for Java 2.x](#).

Documentation et ressources supplémentaires

En plus de ce guide, voici des ressources en ligne utiles pour les développeurs AWS SDK for Java :

- [Référence API AWS SDK for Java](#)
- [Blog des développeurs Java](#)
- [Forums dédiés aux développeurs Java](#)
- GitHub:
 - [Source de documentation](#)
 - [Problèmes de documentation](#)
 - [Source SDK](#)
 - [Problèmes du kit SDK](#)
 - [Exemples du kit SDK](#)
 - [Chaîne Gitter](#)
- L'interface [Catalogue d'exemples de code AWS](#)
- [@awsforjava \(Twitter\)](#)
- [Notes de mise à jour](#)

Prise en charge de l'IDE Eclipse

Si vous développez du code à l'aide de l'IDE Eclipse, vous pouvez utiliser le [AWS Toolkit for Eclipse](#) pour l'ajouter AWS SDK for Java à un projet Eclipse existant ou pour créer un nouveau AWS SDK for Java projet. Cette boîte à outils prend également en charge la création et le chargement de fonctions Lambda, le lancement et la surveillance d'instances Amazon EC2, la gestion des utilisateurs et des groupes de sécurité IAM, un éditeur de modèles AWS CloudFormation, et bien plus encore.

Consultez le [guide de AWS Toolkit for Eclipse l'utilisateur](#) pour obtenir une documentation complète.

Développement d'applications pour Android

Si vous êtes un développeur Android, Amazon Web Services publie un SDK spécialement conçu pour le développement Android : [Amplify Android \(SDK AWS mobile pour Android\)](#).

Affichage de l'historique des révisions du kit SDK

Pour consulter l'historique des versions du kit AWS SDK for Java, notamment les modifications et les services pris en charge selon la version du kit SDK, consultez les [notes de mise à jour](#) de ce kit SDK.

Génération de la documentation de référence Java pour les versions précédentes du kit SDK

La [référence AWS SDK for Java d'API](#) représente la version la plus récente de la version 1.x du SDK. Si vous utilisez une version antérieure de la version 1.x, vous souhaitez peut-être accéder à la documentation de référence du SDK correspondant à la version que vous utilisez.

Le moyen le plus simple de créer la documentation consiste à utiliser l'outil de génération [Maven](#) d'Apache. Commencez par télécharger et installer Maven si vous ne l'avez pas déjà sur votre système, puis utilisez les instructions suivantes pour générer la documentation de référence.

1. Recherchez et sélectionnez la version du SDK que vous utilisez sur la page des [versions](#) du référentiel du SDK sur GitHub.
2. Choisissez le lien zip (la plupart des plateformes, y compris Windows) ou tar .gz (Linux, macOS ou Unix) pour télécharger le SDK sur votre ordinateur.

3. Décompressez l'archive dans un répertoire local.
4. Sur la ligne de commande, accédez au répertoire où vous avez décompressé l'archive et saisissez ce qui suit.

```
mvn javadoc:javadoc
```

5. Une fois la génération terminée, vous trouverez la documentation HTML générée dans le répertoire `aws-java-sdk/target/site/apidocs/`.

Démarrer

Cette section fournit des informations sur l'installation, la configuration et l'utilisation du kit AWS SDK for Java.

Rubriques

- [Configuration de base pour travailler avec Services AWS](#)
- [Moyens d'obtenir le AWS SDK for Java](#)
- [Création d'outils](#)
- [Configurer des informations d'identification AWS temporaires et Région AWS pour le développement](#)

Configuration de base pour travailler avec Services AWS

Présentation

Pour développer correctement des applications qui accèdent à l'Services AWS aide du AWS SDK for Java, les conditions suivantes sont requises :

- Vous devez être en mesure de vous [connecter au portail AWS d'accès](#) disponible dans le AWS IAM Identity Center.
- Les [autorisations du rôle IAM](#) configuré pour le SDK doivent autoriser l'accès à Services AWS ce dont votre application a besoin. Les autorisations associées à la politique PowerUserAccessAWS gérée sont suffisantes pour répondre à la plupart des besoins de développement.
- Un environnement de développement avec les éléments suivants :
 - [Fichiers de configuration partagés](#) qui sont configurés de la manière suivante :
 - Le config fichier contient un profil par défaut qui spécifie un Région AWS.
 - Le credentials fichier contient des informations d'identification temporaires dans le cadre d'un profil par défaut.
 - Une [installation appropriée de Java](#).
 - Un [outil d'automatisation de la création](#) tel que [Maven](#) ou [Gradle](#).
 - Un éditeur de texte pour travailler avec du code.
 - (Facultatif, mais recommandé) Un IDE (environnement de développement intégré) tel qu'[IntelliJ IDEA](#), [Eclipse](#) ou [NetBeans](#).

Lorsque vous utilisez un IDE, vous pouvez également intégrer AWS Toolkit s pour travailler plus facilement avec Services AWS. Les [AWS Toolkit for IntelliJ](#) et [AWS Toolkit for Eclipse](#) sont deux boîtes à outils que vous pouvez utiliser pour le développement Java.

Important

Les instructions de cette section de configuration supposent que vous ou votre organisation utilisez IAM Identity Center. Si votre organisation utilise un fournisseur d'identité externe qui fonctionne indépendamment d'IAM Identity Center, découvrez comment obtenir des informations d'identification temporaires à utiliser dans le SDK for Java. Suivez [ces instructions](#) pour ajouter des informations d'identification temporaires au `~/.aws/credentials` fichier.

Si votre fournisseur d'identité ajoute automatiquement des informations d'identification temporaires au `~/.aws/credentials` fichier, assurez-vous que le nom du profil est `[default]` tel que vous n'avez pas besoin de fournir un nom de profil au SDK ou AWS CLI.

Possibilité de se connecter au portail d'AWS Accès

Le portail AWS d'accès est l'emplacement Web où vous vous connectez manuellement à l'IAM Identity Center. Le format de l'URL est `d-xxxxxxxxxx.awsapps.com/start` ou `your_subdomain.awsapps.com/start`.

Si vous ne connaissez pas le portail d'AWS Accès, suivez les instructions relatives à l'accès au compte décrites à l'[étape 1 de la rubrique sur l'authentification d'IAM Identity Center](#) dans le Guide de référence AWS des kits de développement logiciel et des outils. Ne suivez pas l'étape 2 car la version AWS SDK for Java 1.x ne prend pas en charge l'actualisation automatique des jetons ni la récupération automatique des informations d'identification temporaires pour le SDK décrit à l'étape 2.

Configurer des fichiers de configuration partagés

Les fichiers de configuration partagés résident sur votre poste de travail de développement et contiennent les paramètres de base utilisés par tous les AWS SDK et la AWS Command Line Interface (CLI). Les fichiers de configuration partagés peuvent contenir [un certain nombre de paramètres](#), mais ces instructions définissent les éléments de base requis pour fonctionner avec le SDK.

Configuration du **config** fichier partagé

L'exemple suivant montre le contenu d'un config fichier partagé.

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

À des fins de développement, utilisez le code de la Région AWS [plus proche](#) de l'endroit où vous prévoyez d'exécuter votre code. Pour obtenir la [liste des codes de région](#) à utiliser dans le config fichier, consultez la Référence générale d'Amazon Web Services guide. Le json paramètre du format de sortie est l'une des [nombreuses valeurs possibles](#).

Suivez les instructions [de cette section](#) pour créer le config fichier.

Configurer des informations d'identification temporaires pour le SDK

Une fois que vous avez accès à un rôle Compte AWS et à un rôle IAM via le portail AWS d'accès, configurez votre environnement de développement avec des informations d'identification temporaires auxquelles le SDK pourra accéder.

Étapes pour configurer un **credentials** fichier local avec des informations d'identification temporaires

1. [Créez un credentials fichier partagé.](#)
2. Dans le credentials fichier, collez le texte de l'espace réservé suivant jusqu'à ce que vous y colliez des informations d'identification temporaires valides.

```
[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. Sauvegardez le fichier. Le fichier `~/.aws/credentials` doit maintenant exister sur votre système de développement local. Ce fichier contient le [profil \[par défaut\]](#) que le SDK for Java utilise si aucun profil nommé spécifique n'est spécifié.
4. [Connectez-vous au portail d'AWSaccès.](#)
5. Suivez ces instructions sous la rubrique [Actualisation manuelle des informations d'identification](#) pour copier les informations d'identification du rôle IAM depuis le portail d'AWSaccès.

Choix d'une machine virtuelle Java

Pour que vous obteniez les meilleures performances de vos applications serveur avec le kit AWS SDK for Java, nous vous recommandons d'utiliser la version 64 bits de la machine virtuelle Java. Cette machine virtuelle Java ne s'exécute qu'en mode serveur, même si vous spécifiez l'option `-Client` au moment de l'exécution.

L'utilisation de la version 32 bits de la machine virtuelle Java avec l'option `-Server` au moment de l'exécution doit fournir des performances comparables à la machine virtuelle Java 64 bits.

Moyens d'obtenir le AWS SDK for Java

Prérequis

Pour utiliser le AWS SDK for Java, vous devez avoir :

- Vous devez être en mesure de vous [connecter au portail AWS d'accès](#) disponible dans le AWS IAM Identity Center.
- Une [installation appropriée de Java](#).
- Informations d'identification temporaires configurées dans votre `credentials` fichier partagé local.

Consultez [the section called "Configuration de base"](#) cette rubrique pour savoir comment configurer l'utilisation du SDK for Java.

Utiliser un outil de génération pour gérer les dépendances du SDK for Java (recommandé)

Nous vous recommandons d'utiliser Apache Maven ou Gradle avec votre projet pour accéder aux dépendances requises du SDK for Java. [Cette section](#) décrit comment utiliser ces outils.

Téléchargez et extrayez le SDK (non recommandé)

Nous vous recommandons d'utiliser un outil de génération pour accéder au SDK de votre projet. Vous pouvez toutefois télécharger un fichier jar prédéfini contenant la dernière version du SDK.

Note

Pour plus d'informations sur le téléchargement et la génération des versions précédentes du kit SDK, consultez [Installation des versions précédentes du kit SDK](#).

1. Téléchargez le SDK depuis <https://sdk-for-java.amazonwebservices.com/latest/.zip.aws-java-sdk>
2. Après avoir téléchargé le kit SDK, décompressez le contenu dans un répertoire local.

Le kit SDK contient les répertoires suivants :

- `documentation`- contient la documentation de l'API (également disponible sur le Web : [AWS SDK for Java API Reference](#)).
- `lib`- contient les `.jar` fichiers du SDK.
- `samples`- contient un exemple de code fonctionnel qui montre comment utiliser le SDK.
- `third-party/lib`- contient des bibliothèques tierces utilisées par le SDK, telles que Apache Commons Logging, AspectJ et le framework Spring.

Pour utiliser le kit SDK, ajoutez le chemin d'accès complet des répertoires `lib` et `third-party` aux dépendances de votre fichier de génération, puis ajoutez-les à votre CLASSPATH Java pour exécuter votre code.

Construire les versions précédentes du SDK à partir des sources (non recommandé)

Seule la dernière version du SDK complet est fournie sous forme prédéfinie sous forme de fichier jar téléchargeable. Cependant, vous pouvez générer une version précédente du kit SDK avec Apache Maven (open source). Maven télécharge toutes les dépendances nécessaires, puis génère et installe le kit SDK, le tout en une seule étape. Pour plus d'informations et obtenir les instructions d'installation, consultez <http://maven.apache.org/>.

1. Accédez à la GitHub page du SDK à l'adresse : [AWS SDK for Java \(GitHub\)](#).
2. Choisissez la balise correspondant au numéro de version du kit SDK de votre choix. Par exemple, `1.6.10`.
3. Cliquez sur le bouton Download ZIP (Télécharger le zip) pour télécharger la version du kit SDK que vous avez sélectionnée.

4. Décompressez le fichier dans un répertoire de votre système de développement. Sur de nombreux systèmes, vous pouvez utiliser votre gestionnaire de fichiers graphiques à cette fin, ou utilisez l'utilitaire `unzip` dans une fenêtre de terminal.
5. Dans une fenêtre de terminal, accédez au répertoire où vous avez décompressé le source du kit SDK.
6. Développez et installez le kit SDK avec la commande suivante ([Maven](#) requis) :

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

Le fichier `.jar` généré est intégré au répertoire `target`.

7. (Facultatif) Générez la documentation de référence de l'API à l'aide de la commande suivante :

```
mvn javadoc:javadoc
```

La documentation est intégrée au répertoire `target/site/apidocs/`.

Création d'outils

L'utilisation d'outils de construction permet de gérer le développement de projets Java. Plusieurs outils de génération sont disponibles, mais nous vous expliquons comment vous y prendre en main avec deux outils de construction populaires : Maven et Gradle. Cette rubrique explique comment utiliser ces outils de génération pour gérer le SDK pour les dépendances Java dont vous avez besoin pour vos projets.

Rubriques

- [Utiliser le SDK avec Apache Maven](#)
- [Utiliser le SDK avec Gradle](#)

Utiliser le SDK avec Apache Maven

Vous pouvez utiliser [Apache Maven](#) pour configurer et générer des projets AWS SDK for Java ou pour générer le kit SDK lui-même.

Note

Maven doit être installé sur votre ordinateur pour que vous puissiez utiliser les instructions de cette rubrique. Si tel n'est pas le cas, rendez-vous sur <http://maven.apache.org/> pour le télécharger et l'installer.

Création d'un package Maven

Pour créer un package Maven de base, ouvrez une fenêtre de terminal (ligne de commande) et exécutez :

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

Remplacez `org.example.basicapp` par l'espace de noms complet du package de votre application et `myapp` par le nom de votre projet (celui-ci devient le nom du répertoire de votre projet).

Par défaut, crée un modèle de projet pour vous à l'aide de l'archétype de [démarrage rapide](#), qui constitue un bon point de départ pour de nombreux projets. D'autres archétypes sont disponibles ; rendez-vous sur la page des [archétypes de Maven](#) pour obtenir la liste des archétypes fournis avec. Vous pouvez choisir un archétype particulier à utiliser en ajoutant l'argument `-DarchetypeArtifactId` à la commande `archetype:generate`. Par exemple :

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

Note

Vous trouverez de plus amples informations sur la création et la configuration de projets dans le [guide de démarrage de Maven](#).

Configuration du kit SDK en tant que dépendance Maven

Pour utiliser le kit AWS SDK for Java dans votre projet, vous devez le déclarer comme dépendance dans le fichier `pom.xml` de votre projet. Depuis la version 1.9.0, vous pouvez importer des [composants individuels](#) ou [l'intégralité du kit SDK](#).

Spécification individuelle des modules SDK

Pour sélectionner des modules SDK individuels, utilisez la nomenclature Maven du kit AWS SDK for Java, qui garantit que les modules que vous spécifiez recourent à la même version du kit SDK et qu'ils sont compatibles entre eux.

Pour utiliser la nomenclature, ajoutez une section `<dependencyManagement>` au fichier `pom.xml` de votre application, en ajoutant `aws-java-sdk-bom` en tant que dépendance et en spécifiant la version du kit SDK que vous voulez utiliser :

```
<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>
```

Pour consulter la dernière version de laAWS SDK for Java nomenclature disponible sur Maven Central, rendez-vous sur : <https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-bom>. Vous pouvez également utiliser cette page pour savoir quels sont les modules (dépendances) gérés par la nomenclature et que vous pouvez inclure dans la section `<dependencies>` du fichier `pom.xml` de votre projet.

Vous pouvez maintenant sélectionner individuellement les modules du kit SDK que vous utilisez dans votre application. Dans la mesure où vous avez déjà déclaré la version du kit SDK dans la nomenclature, il n'est pas nécessaire de spécifier le numéro de version de chaque composant.

```
<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>
```

Vous pouvez également vous référer Catalogue d'exemples de code AWS au pour savoir quelles dépendances utiliser pour une donnée Service AWS. Reportez-vous au fichier POM sous un exemple de service spécifique. Par exemple, si vous êtes intéressé par les dépendances du service AWS S3, consultez l'[exemple complet](#) sur GitHub. (Observez le fichier POM sous `/java/example_code/s3`).

Importation de tous les modules SDK

Si vous souhaitez enregistrer l'intégralité du kit SDK comme dépendance, n'utilisez pas la méthode de nomenclature, mais déclarez simplement le kit SDK dans votre fichier `pom.xml`, comme suit :

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

Génération de votre projet

Une fois que vous avez configuré votre projet, vous pouvez le générer à l'aide de la commande Maven `package` :

```
mvn package
```

Cette opération crée votre fichier `0.jar` dans le répertoire `target`.

Génération du kit SDK avec Maven

Vous pouvez utiliser Apache Maven pour générer le kit SDK à partir du code source. Pour ce faire, [téléchargez le code du SDK depuis GitHub](#), décompressez-le localement, puis exécutez la commande Maven suivante :

```
mvn clean install
```

Utiliser le SDK avec Gradle

Pour gérer les dépendances du SDK pour votre [Gradle](#) projet, importez la nomenclature Maven pour AWS SDK for Java dans le `build.gradle` fichier.

Note

Dans les exemples suivants, remplacez `1, 12, 529` dans le fichier de construction contenant une version valide du AWS SDK for Java. Trouvez la dernière version dans le [Référentiel central Maven](#).

Configuration du projet pour Gradle version 4.6 ou ultérieure

[Depuis Gradle 4.6](#), vous pouvez utiliser la fonctionnalité de support POM améliorée de Gradle pour importer des fichiers de nomenclature (BOM) en déclarant une dépendance à une nomenclature.

1. Si vous utilisez Gradle version 5.0 ou ultérieure, passez à l'étape 2. Sinon, activez la fonction `IMPROVED_POM_SUPPORT` dans le fichier `settings.gradle`.

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

2. Ajoutez la nomenclature à la section de dépendances de l'application `build.gradle` fichier.

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

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

3. Spécifiez les modules SDK que vous souhaitez utiliser dans la section des dépendances. L'exemple suivant inclut une dépendance pour Amazon Simple Storage Service (Amazon S3).

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

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

Gradle résout automatiquement la version correcte des dépendances de votre kit SDK à l'aide des informations de la nomenclature.

L'exemple suivant présente un fichier `build.gradle` complet qui inclut une dépendance pour 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

Dans l'exemple précédent, remplacez la dépendance pour Amazon S3 avec les dépendances du AWS services que vous utiliserez dans le cadre de votre projet. Les modules (dépendances) gérés par AWS SDK for Java Les BOM sont listés sur [Référentiel central Maven](#).

Configuration de projet pour les versions de Gradle antérieures à la version 4.6

Les versions de Gradle antérieures à la version 4.6 ne prennent pas en charge la nomenclature native. Pour gérer les dépendances du AWS SDK for Java de votre projet, utilisez le [plugin de gestion des dépendances](#) de Spring pour Gradle afin d'importer la nomenclature Maven pour le kit SDK.

1. Ajoutez le plugin de gestion des dépendances à celui de votre application `build.gradle` fichier.

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

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

2. Ajoutez la nomenclature dans la section `dependencyManagement` du fichier.

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

3. Spécifiez les modules SDK que vous utiliserez dans la section des dépendances. L'exemple suivant inclut une dépendance pour Amazon S3.

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

Gradle résout automatiquement la version correcte des dépendances de votre kit SDK à l'aide des informations de la nomenclature.

L'exemple suivant présente un fichier `build.gradle` complet qui inclut une dépendance pour 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

Dans l'exemple précédent, remplacez la dépendance pour Amazon S3 avec les dépendances du AWS service que vous utiliserez dans votre projet. Les modules (dépendances) gérés par AWS SDK for Java Les BOM sont listés sur [Référentiel central Maven](#).

Pour plus de détails sur la spécification des dépendances du kit SDK à l'aide de la nomenclature, consultez [Utilisation du kit SDK avec Apache Maven](#).

Configurer des informations d'identification AWS temporaires et Région AWS pour le développement

Pour vous connecter à l'un des services pris en charge avec le AWS SDK for Java, vous devez fournir des informations d'identification AWS temporaires. Les AWS kits SDK et les interface de ligne de commande utilisent les chaînes fournisseur pour rechercher les informations d'identification AWS temporaires dans un certain nombre d'emplacements différents, y compris les variables d'environnement système ou utilisateur et les fichiers AWS de configuration locaux.

Cette rubrique fournit des informations de base sur la configuration de vos informations d'identification AWS temporaires pour le développement d'applications locales à l'aide du AWS SDK for Java. Si vous devez configurer des informations d'identification à utiliser dans une instance EC2 ou si vous utilisez l'IDE Eclipse pour le développement, veuillez plutôt consulter les rubriques suivantes :

- Lorsque vous utilisez une instance EC2, créez un rôle IAM, puis accordez à votre instance EC2 l'accès à ce rôle, comme indiqué dans la [section Utilisation des rôles IAM pour accorder l'accès aux AWS ressources sur Amazon EC2](#).
- Configurez les AWS informations d'identification dans Eclipse à l'aide du [AWS Toolkit for Eclipse](#). Reportez-vous à la section [Configurer les AWS informations d'identification](#) dans le [guide de AWS Toolkit for Eclipse l'utilisateur](#) pour plus d'informations.

Configurer les informations d'identification temporaires sous

Vous pouvez configurer des informations d'identification temporaires pour le AWS SDK for Java de différentes manières, mais voici les approches recommandées :

- Définissez des informations d'identification temporaires dans le fichier de profil des AWS informations d'identification de votre système local, situé à l'adresse suivante :
 - `~/.aws/credentials` sous Linux, macOS ou Unix
 - `C:\Users\USERNAME\.aws\credentials` sous Windows

Consultez ce guide pour obtenir des instructions sur la façon d'obtenir vos informations d'identification temporaires. [the section called “Configurer des informations d'identification temporaires pour le SDK”](#)

- Définissez `AWS_ACCESS_KEY_ID` les variables d'`AWS_SESSION_TOKEN` environnement `AWS_SECRET_ACCESS_KEY`, et.

Pour définir ces variables sous Linux, macOS ou Unix, utilisez :

```
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
```

Pour définir ces variables sous Windows, utilisez :

```
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
```

- Pour une instance EC2, spécifiez un rôle IAM, puis autorisez l'instance EC2 à y accéder. Reportez-vous à la section [Rôles IAM Amazon EC2](#) dans le Guide de Amazon EC2 l'utilisateur pour les instances Linux pour une discussion détaillée sur la façon dont cela fonctionne.

Une fois que vous avez défini vos informations d'identification AWS temporaires à l'aide de l'une de ces méthodes, elles seront chargées automatiquement AWS SDK for Java par le à l'aide de la chaîne de fournisseurs d'informations d'identification par défaut. Pour plus d'informations sur l'utilisation des AWS informations d'identification dans vos applications Java, consultez la section [Utilisation des AWS informations d'identification](#).

Actualisation des informations d'identification de l'IMDS

Le kit AWS SDK for Java prend en charge l'actualisation de l'inscription des informations d'identification de l'IMDS en arrière-plan toutes les minutes, quelle que soit la date d'expiration des informations d'identification. Cela vous permet d'actualiser les informations d'identification plus fréquemment et de réduire le risque que le fait de ne pas accéder à IMDS ait une incidence sur la AWS disponibilité perçue.

```
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();
```

Réglez la Région AWS

Vous devez définir une valeur par défaut Région AWS qui sera utilisée pour accéder aux AWS services avec le AWS SDK for Java. Pour des performances réseau optimales, choisissez une région qui est géographiquement proche de chez vous (ou de vos clients). Pour obtenir la liste des régions pour chaque service, voir [Régions et points de terminaison](#) dans la Référence Amazon Web Services générale.

Note

Si vous ne sélectionnez aucune région, us-east-1 sera utilisé par défaut.

Vous pouvez utiliser des techniques similaires à la définition des informations d'identification pour définir votre AWS région par défaut :

- Définissez la Région AWS dans le fichier de configuration de votre système local, situé à l'adresse suivante :
 - ~/.....sous Linux, macOS ou Unix sous Linux, macOS ou Unix sous Linux
 - C:\Users\USERNAME \ .aws \ config sous Windows

Ce fichier doit contenir des lignes au format suivant :

+

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

+

Remplacez votre région souhaitée Région AWS (par exemple, « us-east-1 ») par `your_aws_region`.

- Définissez la variable d'environnement `AWS_REGION`.

Sous Linux, macOS ou Unix sous Linux, macOS ou Unix sous Linux, macOS ou Unix

```
export AWS_REGION=your_aws_region
```

Sous Windows, utilisez :

```
set AWS_REGION=your_aws_region
```

Où `your_aws_region` est le Région AWS nom souhaité.

À l'aide du AWS SDK for Java

Cette section fournit des informations générales importantes sur la programmation avec le AWS SDK for Java qui s'appliquent à tous les services que vous pouvez utiliser avec le SDK.

Pour obtenir des informations et des exemples de programmation spécifiques au service (pour Amazon EC2, Amazon S3, Amazon SWF, etc.), voir Exemples de [AWS SDK for Java code](#).

Rubriques

- [Meilleures pratiques en matière AWS de développement avec AWS SDK for Java](#)
- [Création de clients de service](#)
- [Fournissez des informations d'identification temporaires au AWS SDK for Java](#)
- [Région AWS Sélection](#)
- [Gestion des exceptions](#)
- [Programmation asynchrone](#)
- [Enregistrement AWS SDK for Java des appels](#)
- [Configuration de client](#)
- [Stratégies de contrôle d'accès](#)
- [Définissez le TTL de la JVM pour les recherches de noms DNS](#)
- [Activation des métriques pour AWS SDK for Java](#)

Meilleures pratiques en matière AWS de développement avec AWS SDK for Java

Les meilleures pratiques suivantes peuvent vous aider à éviter les problèmes lorsque vous développez des AWS applications avec le AWS SDK for Java. Nous avons organisé les bonnes pratiques par service.

S3

Éviter ResetExceptions

Lorsque vous chargez des objets à Amazon S3 l'aide de flux (via un AmazonS3 client ou `TransferManager`), vous pouvez rencontrer des problèmes de connectivité réseau ou de délai

d'expiration. Par défaut, les AWS SDK for Java tentatives de nouvelle tentative de transfert ont échoué en marquant le flux d'entrée avant le début du transfert, puis en le réinitialisant avant de réessayer.

Si le flux ne prend pas en charge le marquage et la réinitialisation, le SDK lance un message en [ResetException](#) cas d'échec transitoire et les nouvelles tentatives sont activées.

Bonne pratique

Nous vous recommandons d'utiliser des flux qui prennent en charge les opérations de marquage et de réinitialisation.

Le moyen le plus fiable d'éviter un [ResetException](#) est de fournir des données à l'aide d'un [fichier](#) ou [FileInputStream](#), qu'ils AWS SDK for Java peuvent gérer sans être limités par des limites de marquage et de réinitialisation.

Si le stream n'est pas un [FileInputStream](#) mais qu'il prend en charge le marquage et la réinitialisation, vous pouvez définir la limite de points en utilisant la `setReadLimit` méthode de [RequestClientOptions](#). Sa valeur par défaut est 128 Ko. La définition de la valeur limite de lecture à un octet de plus que la taille du flux évitera de manière fiable un [ResetException](#).

Par exemple, si la taille maximale attendue d'un flux est 100 000 octets, définissez la limite de lecture sur 100 001 (100 000 + 1) octets. Le marquage et la réinitialisation fonctionneront toujours pour 100 000 octets ou moins. Ayez à l'esprit que cela peut entraîner le fait que certains flux mettent en mémoire tampon le nombre d'octets en mémoire.

Création de clients de service

Pour envoyer des demandes à Amazon Web Services, vous devez d'abord créer un objet client de service. La méthode recommandée consiste à utiliser le générateur client de service.

Chacun Service AWS possède une interface de service avec des méthodes pour chaque action dans l'API de service. [Par exemple, l'interface de service de DynamoDB s'appelle DbClient.](#) [AmazonDynamo](#) Chaque interface de service possède un générateur client correspondant que vous pouvez utiliser pour construire une implémentation de l'interface de service. La classe de création de clients pour DynamoDB est nommée [AmazonDynamoDB ClientBuilder](#).

Obtention d'un générateur client

Pour obtenir une instance du générateur client, utilisez la méthode de fabrique statique standard, comme illustré dans l'exemple suivant.

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

Une fois que vous disposez d'un générateur, vous pouvez personnaliser les propriétés du client à l'aide de nombreuses méthodes setter Fluent dans l'API du générateur. Par exemple, vous pouvez définir une région personnalisée et un fournisseur d'informations d'identification personnalisé, comme suit.

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

Note

Les méthodes withXXX Fluent renvoient l'objet builder pour vous permettre de chaîner les appels de méthode pour plus de commodité et un code plus lisible. Une fois que vous avez configuré les propriétés souhaitées, vous pouvez appeler la méthode build pour créer le client. Une fois qu'un client a été créé, il est immuable et tous les appels à setRegion ou setEndpoint échoueront.

Un générateur peut créer plusieurs clients avec la même configuration. Lorsque vous écrivez votre application, gardez à l'esprit que le générateur est mutable et n'est pas thread-safe.

Le code suivant utilise le générateur en tant que fabrique pour les instances client.

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

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



```
}  
}
```

[Le générateur expose également des setters fluides pour ClientConfiguration et RequestMetricCollector, ainsi qu'une liste personnalisée de RequestHandler 2.](#)

Voici un exemple complet qui remplace toutes les propriétés configurables.

```
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();
```

Création de clients asynchrones

AWS SDK for Java II possède des clients asynchrones (ou asynchrones) pour chaque service (sauf pour Amazon S3) et un générateur de clients asynchrones correspondant pour chaque service.

Pour créer un client DynamoDB asynchrone avec la valeur par défaut ExecutorService

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

Outre les options de configuration prises en charge par le générateur de clients synchrones (ou de synchronisation), le client asynchrone vous permet de définir une personnalisation [ExecutorFactory](#) pour modifier ExecutorService celle utilisée par le client asynchrone.

ExecutorFactory est une interface fonctionnelle, elle interagit donc avec les expressions lambda de Java 8 et les références de méthodes.

Pour créer un client asynchrone avec un exécuteur personnalisé

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

En utilisant DefaultClient

Les générateurs client synchrones et asynchrones ont une autre méthode nommée `defaultClient`. Cette méthode crée un client de service avec la configuration par défaut, en utilisant la chaîne de fournisseurs par défaut pour charger les informations d'identification et le Région AWS. Si les informations d'identification ou la région ne peuvent pas être déterminées à partir de l'environnement dans lequel l'application s'exécute, l'appel à `defaultClient` échoue. Consultez la section [Utilisation des AWS informations d'identification](#) et [Région AWS sélection](#) pour plus d'informations sur la manière dont les informations d'identification et la région sont déterminées.

Pour créer un service client par défaut

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

Cycle de vie des clients

Les clients de service du kit SDK sont thread-safe, et pour des performances optimales, vous devrez les traiter comme des objets à longue durée de vie. Chaque client possède sa propre ressource de groupe de connexion. Arrêtez explicitement les clients lorsque vous n'en avez plus besoin pour éviter les fuites de ressources.

Pour arrêter explicitement un client, appelez la méthode `shutdown`. Une fois la méthode `shutdown` appelée, toutes les ressources client sont libérées et le client devient inutilisable.

Pour arrêter un client

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

Fournissez des informations d'identification temporaires au AWS SDK for Java

Pour faire des demandes à Amazon Web Services, vous devez fournir des informations d'identification AWS temporaires AWS SDK for Java à utiliser lorsqu'il appelle les services. Vous pouvez effectuer cette opération de différentes manières :

- Utilisez la chaîne de fournisseur d'informations d'identification par défaut (recommandé).

- Utilisez un fournisseur ou une chaîne de fournisseur d'informations d'identification spécifique (ou créez le vôtre).
- Fournissez vous-même les informations d'identification temporaires sous forme de code.

Utilisation de la chaîne de fournisseur d'informations d'identification par défaut

Lorsque vous initialisez un nouveau client de service sans fournir d'arguments, il AWS SDK for Java tente de trouver des informations d'identification temporaires en utilisant la chaîne de fournisseurs d'informations d'identification par défaut implémentée par la classe [AWSCredentialsProviderChainDefault](#). La chaîne de fournisseur d'informations d'identification par défaut recherche les informations d'identification dans l'ordre suivant :

1. Variables d'environnement - `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, et `AWS_SESSION_TOKEN`. AWS SDK for Java Utilise la [EnvironmentVariableCredentialsProvider](#) classe pour charger ces informations d'identification.
2. Propriétés du système Java - `aws.accessKeyId`, `aws.secretKey`, et `aws.sessionToken`. AWS SDK for Java Utilise le [SystemPropertiesCredentialsProvider](#) pour charger ces informations d'identification.
3. Informations d'identification du jeton d'identité web à partir de l'environnement ou du conteneur.
4. Le fichier de profils d'identification par défaut, généralement situé dans `~/.aws/credentials` (l'emplacement peut varier selon la plate-forme) et partagé par de nombreux AWS SDK et par le AWS CLI. AWS SDK for Java Utilise le [ProfileCredentialsProvider](#) pour charger ces informations d'identification.

Vous pouvez créer un fichier d'informations d'identification à l'aide de la `aws configure` commande fournie par le AWS CLI, ou vous pouvez le créer en modifiant le fichier à l'aide d'un éditeur de texte. Pour plus d'informations sur le format de fichier d'informations d'identification, voir [Format de fichier AWS d'informations d'identification](#).

5. Informations d'identification du conteneur Amazon ECS : chargées depuis Amazon ECS si la variable d'environnement `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` est définie. AWS SDK for Java Utilise le [ContainerCredentialsProvider](#) pour charger ces informations d'identification. Vous pouvez spécifier l'adresse IP de cette valeur.
6. Informations d'identification du profil d'instance : utilisées sur les instances EC2 et fournies via le service de Amazon EC2 métadonnées. AWS SDK for Java Utilise le

[InstanceProfileCredentialsProvider](#) pour charger ces informations d'identification. Vous pouvez spécifier l'adresse IP de cette valeur.

Note

Les informations d'identification de profil d'instance sont utilisées uniquement si la variable d'environnement `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` n'est pas définie. Consultez [EC2 ContainerCredentialsProviderWrapper](#) pour plus d'informations.

Définissez des informations d'identification temporaires

Pour pouvoir utiliser des informations d'identification AWS temporaires, elles doivent être définies dans au moins l'un des emplacements précédents. Pour plus d'informations sur la définition des informations d'identification, consultez les rubriques suivantes :

- Pour spécifier les informations d'identification dans l'environnement ou dans le fichier de profils d'identification par défaut, consultez [the section called “Configurer les informations d'identification temporaires sous”](#).
- Pour définir les propriétés système Java, consultez le didacticiel [System Properties](#) sur le site web Java Tutorials officiel.
- Pour configurer et utiliser les informations d'identification du profil d'instance avec vos instances EC2, consultez la section [Utilisation des rôles IAM pour accorder l'accès aux AWS ressources sur Amazon EC2](#)

Définir un autre profil d'identification

AWS SDK for Java Utilise le profil par défaut, mais il existe des moyens de personnaliser le profil qui provient du fichier d'informations d'identification.

Vous pouvez utiliser la variable d'environnement `AWS Profile` pour modifier le profil chargé par le SDK.

Par exemple, sous Linux, macOS ou Unix, vous devez exécuter la commande suivante pour remplacer le profil par `MyProfile`.

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

Sous Windows, utilisez la commande suivante.

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

La définition de la variable d'AWS_PROFILE environnement affecte le chargement des informations d'identification pour tous les AWS SDK et outils officiellement pris en charge (y compris le AWS CLI et le AWS Tools for Windows PowerShell). Pour modifier uniquement le profil d'une application Java, vous pouvez utiliser la propriété système à la `aws.profile` place.

Note

La variable d'environnement est prioritaire sur la propriété système.

Définir un autre emplacement pour le fichier d'informations d'identification

AWS SDK for Java Charge automatiquement les informations d'identification AWS temporaires à partir de l'emplacement du fichier d'informations d'identification par défaut. Cependant, vous pouvez également spécifier l'emplacement en définissant la variable d'environnement `AWS_CREDENTIAL_PROFILES_FILE` avec le chemin d'accès complet au fichier d'informations d'identification.

Vous pouvez utiliser cette fonctionnalité pour modifier temporairement l'emplacement où AWS SDK for Java recherche votre fichier d'informations d'identification (par exemple, en définissant cette variable avec la ligne de commande). Vous pouvez également définir la variable d'environnement dans votre environnement utilisateur ou système pour modifier l'emplacement pour l'utilisateur ou l'ensemble du système.

Pour remplacer l'emplacement du fichier d'informations d'identification par défaut

- Définissez la variable d'AWS_CREDENTIAL_PROFILES_FILE environnement sur l'emplacement de votre fichier AWS d'informations d'identification.
 - Sous Linux, macOS ou Unix, utilisez :

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

- Sous Windows, utilisez :

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

Credentials format de fichier

En suivant les [instructions de la section Configuration de base](#) de ce guide, votre fichier d'informations d'identification doit avoir le format de base suivant.

```
[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>
```

Le nom de profil est spécifié entre crochets (par exemple, [default]), suivi par les champs configurables de ce profil sous la forme de paires clé-valeur. Vous pouvez avoir plusieurs profils dans votre credentials fichier, qui peuvent être ajoutés ou modifiés en `aws configure --profile PROFILE_NAME` sélectionnant le profil à configurer.

Vous pouvez spécifier des champs supplémentaires, tels que `metadata_service_timeout`, `metadata_service_num_attempts`. Ils ne sont pas configurables avec la CLI : vous devez modifier le fichier manuellement si vous souhaitez les utiliser. Pour plus d'informations sur le fichier de configuration et les champs disponibles, consultez [la section Configuration du AWS Command Line Interface](#) dans le guide de AWS Command Line Interface l'utilisateur.

Charger les identifiants

Une fois que vous avez défini des informations d'identification temporaires, le SDK les charge en utilisant la chaîne de fournisseurs d'informations d'identification par défaut.

Pour ce faire, vous instanciez un Service AWS client sans fournir explicitement d'informations d'identification au générateur, comme suit.

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

Spécifiez un fournisseur d'informations d'identification ou une chaîne de fournisseurs

Vous pouvez spécifier un fournisseur d'informations d'identification autre que la chaîne de fournisseur d'informations d'identification par défaut à l'aide du générateur client.

Vous fournissez une instance d'un fournisseur d'informations d'identification ou d'une chaîne de fournisseurs à un générateur de clients qui prend une [AWSCredentialsProvider](#) interface en entrée. L'exemple suivant montre comment utiliser des informations d'identification d'environnement.

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

Pour la liste complète des fournisseurs d'informations d'identification et des chaînes de fournisseurs AWS SDK for Java fournis, voir Toutes les classes d'implémentation connues dans [AWSCredentialsProvider](#)

Note

Vous pouvez utiliser cette technique pour fournir des fournisseurs d'informations d'identification ou des chaînes de fournisseurs que vous créez en utilisant votre propre fournisseur d'informations d'identification qui implémente l'[AWSCredentialsProvider](#) interface, ou en sous-classant la classe.

[AWSCredentialsProviderChain](#)

Spécifiez explicitement les informations d'identification temporaires

Si la chaîne d'informations d'identification par défaut, ou un fournisseur ou une chaîne de fournisseur spécifique ou personnalisé ne fonctionne pas pour votre code, vous pouvez définir des informations d'identification que vous spécifiez explicitement. Si vous avez récupéré des informations d'identification temporaires à l'aide de AWS STS, utilisez cette méthode pour spécifier les informations d'identification pour AWS l'accès.

1. Instanciez la [BasicSessionCredentials](#) classe et fournissez-lui la clé d' AWS accès, la clé AWS secrète et le jeton de AWS session que le SDK utilisera pour la connexion.
2. Créez un [AWSStaticCredentialsProvider](#) avec l'`AWSCredentials` objet.

3. Configurez le générateur client avec l'interface `AWSStaticCredentialsProvider` et générez le client.

Voici un exemple.

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

Plus d'informations

- [Inscrivez-vous AWS et créez un utilisateur IAM](#)
- [Configurer les AWS informations d'identification et la région pour le développement](#)
- [Utilisation des rôles IAM pour accorder l'accès aux AWS ressources sur Amazon EC2](#)

Région AWS Sélection

Les régions vous permettent d'accéder à AWS des services qui résident physiquement dans une zone géographique spécifique. Cela peut être utile pour la redondance et pour maintenir vos données et vos applications en cours d'exécution à proximité de l'endroit où vous-même et vos utilisateurs y accédez.

Vérification de la disponibilité du service dans une région

Pour savoir si un produit spécifique Service AWS est disponible dans une région, utilisez la `isServiceSupported` méthode de la région que vous souhaitez utiliser.

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

Consultez la documentation sur la classe [Regions](#) pour les régions que vous pouvez spécifier, et utilisez le préfixe de point de terminaison du service à interroger. Chaque préfixe de point de terminaison du service est défini dans l'interface du service. Par exemple, le préfixe du DynamoDB point de terminaison est défini dans la [AmazonDynamobase de données](#).

Choisir une région

À partir de la version 1.4 du AWS SDK for Java, vous pouvez spécifier un nom de région et le SDK choisira automatiquement un point de terminaison approprié pour vous. Pour choisir le point de terminaison vous-même, consultez [Choix d'un point de terminaison spécifique](#).

Pour définir explicitement une région, nous vous recommandons d'utiliser l'énumération [Regions](#). Il s'agit d'une énumération de toutes les régions disponibles publiquement. Pour créer un client avec une région à partir de l'énumération, utilisez le code suivant.

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

Si la région que vous essayez d'utiliser n'est pas dans l'énumération `Regions`, vous pouvez définir la région à l'aide d'une chaîne représentant le nom de la région.

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

Note

Une fois que vous avez créé un client avec le générateur, il est immuable et la région ne peut pas être modifiée. Si vous travaillez avec plusieurs clients `Regions` AWS pour le même service, vous devez créer plusieurs clients, un par région.

Choix d'un point de terminaison spécifique

Chaque AWS client peut être configuré pour utiliser un point de terminaison spécifique dans une région en appelant la `withEndpointConfiguration` méthode lors de la création du client.

Par exemple, pour configurer le Amazon S3 client afin qu'il utilise la région Europe (Irlande), utilisez le code suivant.

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

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

Voir [Régions et points de terminaison](#) pour la liste actuelle des régions et leurs points de terminaison correspondants pour tous les AWS services.

Déterminer automatiquement la région à partir de l'environnement

Important

Cette section s'applique uniquement lorsque vous utilisez un [générateur de clients](#) pour accéder aux AWS services. AWS les clients créés à l'aide du constructeur client ne détermineront pas automatiquement la région à partir de l'environnement et utiliseront à la place la région du SDK par défaut (UseAst1).

Lorsque vous exécutez Lambda Amazon EC2 ou Lambda, vous souhaitez peut-être configurer les clients pour qu'ils utilisent la même région que celle sur laquelle votre code s'exécute. Votre code est ainsi dissocié de l'environnement d'exécution et le déploiement de votre application sur plusieurs régions dans le but de réduire la latence ou la redondance s'en trouve simplifié.

Vous devez utiliser les générateurs clients pour que le kit SDK détecte automatiquement la région dans laquelle votre code s'exécute.

Pour utiliser la chaîne du fournisseur d'informations d'identification/de région par défaut et déterminer la région à partir de l'environnement, utilisez la méthode `defaultClient` du générateur client.

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

Cette solution est identique à l'utilisation de `standard` suivi par `build`.

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

Si vous ne définissez pas explicitement une région à l'aide des méthodes `withRegion`, le kit SDK consulte la chaîne du fournisseur de région par défaut afin d'essayer de déterminer la région à utiliser.

Chaîne du fournisseur de région par défaut

Le processus de recherche d'une région est le suivant :

1. Toute région explicite définie en utilisant `withRegion` ou `setRegion` sur le générateur lui-même prévaut sur toute autre région.
2. La variable d'environnement `AWS_REGION` est contrôlée. Si elle est définie, cette région est utilisée pour configurer le client.

Note

Cette variable d'environnement est définie par le Lambda conteneur.

3. Le SDK vérifie le fichier de configuration AWS partagé (généralement situé à l'adresse `~/ .aws/ config`). Si la propriété `region` est présente, le kit SDK l'utilise.
 - La variable d'environnement `AWS_CONFIG_FILE` peut être utilisée pour personnaliser l'emplacement du fichier de configuration partagé.
 - La variable d'environnement `AWS_PROFILE` ou la propriété `aws .profile` système peuvent être utilisées pour personnaliser le profil chargé par le SDK.
4. Le SDK tente d'utiliser le service de métadonnées d' Amazon EC2 instance pour déterminer la région de l' Amazon EC2 instance en cours d'exécution.
5. Si le kit SDK n'a toujours pas trouvé de région par ce biais, la création du client échoue et une exception est levée.

Lors du développement d' AWS applications, une approche courante consiste à utiliser le fichier de configuration partagé (décrit dans [Utilisation de la chaîne de fournisseurs d'informations d'identification par défaut](#)) pour définir la région pour le développement local, et à s'appuyer sur la chaîne de fournisseurs de régions par défaut pour déterminer la région lors de l'exécution sur une AWS infrastructure. La création du client s'en trouve ainsi grandement simplifiée et votre application demeure portable.

Gestion des exceptions

Il est important de comprendre comment et quand AWS SDK for Java les exceptions sont générées pour créer des applications de haute qualité à l'aide du SDK. Les sections suivantes décrivent les différents cas d'exceptions levées par le kit SDK et la manière de les gérer de manière appropriée.

Pourquoi des exceptions non contrôlées ?

AWS SDK for Java Utilise des exceptions d'exécution (ou non vérifiées) au lieu d'exceptions vérifiées pour les raisons suivantes :

- Permettre aux développeurs un contrôle extrêmement précis des erreurs qu'ils veulent gérer sans les forcer à gérer les cas exceptionnels par lesquels ils ne sont pas concernés (rendant alors leur code excessivement détaillé)
- Pour éviter les problèmes d'évolutivité inhérents aux exceptions contrôlées dans les grandes applications

En général, les exceptions contrôlées fonctionnent bien à petite échelle, mais peuvent devenir problématiques au fur et à mesure que les applications se développent et deviennent plus complexes.

Pour plus d'informations sur l'utilisation des exceptions contrôlées et des exceptions non contrôlées, consultez :

- [Les exceptions incontrôlées : la controverse](#)
- [Problème des exceptions contrôlées](#)
- [Les exceptions contrôlées de Java étaient une erreur \(et voici ce que je voudrais faire à ce sujet\)](#)

AmazonServiceException (et sous-classes)

[AmazonServiceException](#) est l'exception la plus courante que vous rencontrerez lors de l'utilisation du AWS SDK for Java. Cette exception représente une réponse d'erreur provenant d'un Service AWS. Par exemple, si vous essayez de mettre fin à une Amazon EC2 instance qui n'existe pas, EC2 renverra une réponse d'erreur et tous les détails de cette réponse d'erreur seront inclus dans le `AmazonServiceException` message envoyé. Dans certains cas, une sous-classe d'`AmazonServiceException` est levée afin de permettre aux développeurs un contrôle très précis de la gestion des cas d'erreur par le biais de blocs d'interception (catch).

Lorsque vous rencontrez un `AmazonServiceException`, vous savez que votre demande a été envoyée avec succès au Service AWS mais n'a pas pu être traitée avec succès. Cela peut être dû à une erreur des paramètres de la demande ou à un problème côté service.

`AmazonServiceException` vous fournit des informations telles que :

- Code d'état HTTP retourné
- Code AWS d'erreur renvoyé
- Message d'erreur détaillé du service
- AWS ID de demande pour la demande qui a échoué

`AmazonServiceException` inclut également des informations indiquant si l'échec de la demande est la faute de l'appelant (demande avec des valeurs illégales) ou la faute Service AWS de l'appelant (erreur de service interne).

AmazonClientException

[AmazonClientException](#) indique qu'un problème s'est produit dans le code du client Java, soit lors de la tentative d'envoi d'une demande, AWS soit lors de la tentative d'analyse d'une réponse de AWS. Un `AmazonClientException` est généralement plus grave qu'un `AmazonServiceException` et indique un problème majeur qui empêche le client de faire des appels de service aux AWS services. Par exemple, il AWS SDK for Java lance une alerte `AmazonClientException` si aucune connexion réseau n'est disponible lorsque vous essayez d'appeler une opération sur l'un des clients.

Programmation asynchrone

Vous pouvez utiliser des méthodes synchrones ou asynchrones pour appeler des opérations sur des services. AWS Les méthodes synchrones bloquent l'exécution du thread jusqu'à ce que le client reçoive une réponse du service. Les méthodes asynchrones renvoient immédiatement, en rendant le contrôle au thread appelant sans attendre de réponse.

Dans la mesure où une méthode asynchrone renvoie avant qu'une réponse ne soit disponible, vous avez besoin d'une solution pour obtenir la réponse quand elle est prête. AWS SDK for Java II propose deux méthodes : les objets futurs et les méthodes de rappel.

Objets Future Java

Les méthodes asynchrones AWS SDK for Java renvoient un objet [Future](#) contenant les résultats de l'opération asynchrone à venir.

Appelez la méthode `Future isDone()` pour voir si le service a déjà fourni un objet de réponse. Lorsque la réponse est prête, vous pouvez obtenir l'objet de la réponse en appelant la méthode

`Future.get()`. Vous pouvez utiliser ce mécanisme pour interroger régulièrement les résultats de l'opération asynchrone, tandis que votre application continue à travailler sur d'autres éléments.

Voici un exemple d'opération asynchrone qui appelle une Lambda fonction et reçoit un objet `Future` pouvant contenir un [InvokeResult](#) objet. L'objet `InvokeResult` est récupéré uniquement après qu'`isDone()` a la valeur `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);
}
```

Rappels asynchrones

Outre l'utilisation de l'`Future` objet Java pour surveiller l'état des demandes asynchrones, le SDK vous permet également d'implémenter une classe qui utilise l'interface [AsyncHandler](#). `AsyncHandler` fournit deux méthodes qui sont appelées en fonction de la manière dont la demande est terminée : `onSuccess` et `onError`.

Le principal avantage de l'approche de l'interface de rappel est qu'il vous évite d'avoir à interroger l'objet `Future` pour savoir à quel moment la demande est terminée. Au lieu de cela, votre code peut immédiatement commencer son activité suivante et s'appuyer sur le kit SDK pour appeler votre gestionnaire au bon moment.

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


Bonnes pratiques

Exécution des rappels

Votre implémentation de `AsyncHandler` est exécutée à l'intérieur du groupe de threads dont le client asynchrone est propriétaire. Un code bref, rapidement exécuté est le plus approprié à l'intérieur de votre implémentation d' `AsyncHandler`. Un code de longue durée ou un code de blocage à l'intérieur des méthodes de votre gestionnaire peuvent entraîner un conflit au sein du groupe de threads utilisé par le client asynchrone et empêcher le client d'exécuter les demandes. Si vous avez une tâche de longue durée qui doit commencer à partir d'un rappel, faites en sorte que le rappel exécute sa tâche dans un nouveau thread ou dans un groupe de threads géré par votre application.

Configuration du groupe de threads

Les clients asynchrones du AWS SDK for Java fournissent un pool de threads par défaut qui devrait fonctionner pour la plupart des applications. Vous pouvez implémenter une personnalisation [ExecutorService](#) et la transmettre à des clients AWS SDK for Java asynchrones pour mieux contrôler la façon dont les pools de threads sont gérés.

Par exemple, vous pouvez fournir une `ExecutorService` implémentation qui utilise un paramètre personnalisé [ThreadFactory](#) pour contrôler le nom des threads du pool ou pour enregistrer des informations supplémentaires sur l'utilisation des threads.

Accès asynchrone

La [TransferManager](#) classe du SDK offre un support asynchrone pour travailler avec. Amazon `S3TransferManager` gère les chargements et téléchargements asynchrones, fournit des rapports d'avancement détaillés sur les transferts et prend en charge les rappels lors de différents événements.

Enregistrement AWS SDK for Java des appels

AWS SDK for Java 1.1 est instrumenté avec [Apache Commons Logging](#), une couche d'abstraction qui permet d'utiliser l'un des nombreux systèmes de journalisation au moment de l'exécution.

Les systèmes de journalisation pris en charge incluent Java Logging Framework et Apache Log4j, entre autres. Cette section vous explique comment utiliser Log4j. Vous pouvez utiliser la fonction de journalisation du kit SDK sans apporter de modifications au code de votre application.

Pour en savoir plus sur [Log4j](#), consultez le [site web Apache](#).

Note

Cette rubrique se concentre sur Log4j 1.x. Log4j2 ne prend pas directement en charge Apache Commons Logging, mais fournit un adaptateur qui dirige automatiquement la journalisation des appels vers Log4j2 à l'aide de l'interface Apache Commons Logging. Pour plus d'informations, consultez [Commons Logging Bridge](#) dans la documentation Log4j2.

Téléchargement du fichier JAR Log4J

Pour utiliser Log4j avec le kit SDK, vous devez télécharger le fichier JAR Log4j à partir du site web Apache. Le kit SDK n'inclut pas le fichier JAR. Copiez le fichier JAR sur un emplacement de votre chemin de classe.

Log4j utilise un fichier de configuration, `log4j.properties`. Vous trouverez ci-dessous des exemples de fichiers de configuration. Copiez ce fichier de configuration dans un répertoire de votre chemin de classe. Le fichier JAR Log4j et le fichier `log4j.properties` ne doivent pas nécessairement se trouver dans le même répertoire.

Le fichier de configuration `log4j.properties` spécifie les propriétés telles que le [niveau de journalisation](#), l'emplacement vers lequel la sortie de la journalisation est envoyée (par exemple, [vers un fichier ou vers la console](#)) et le [format de la sortie](#). Le niveau de journalisation correspond à la granularité de la sortie que l'enregistreur d'événements génère. Log4j prend en charge le concept de hiérarchies de journalisation multiples. Le niveau de journalisation est défini de manière indépendante pour chaque hiérarchie. Les deux hiérarchies de journalisation suivantes sont disponibles dans le kit AWS SDK for Java :

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

Définition du chemin de classe

Le fichier JAR Log4j et le fichier `log4j.properties` doivent se trouver sur votre chemin de classe. Si vous utilisez [Apache Ant](#), définissez le chemin de classe dans l'élément `path` de votre fichier Ant. L'exemple suivant montre un élément de chemin du fichier Ant pour l' Amazon S3 [exemple](#) inclus dans le SDK.

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

Si vous utilisez l'IDE Eclipse, vous pouvez définir le chemin de classe en ouvrant le menu et en accédant à Project (Projet) | Properties (Propriétés) | Java Build Path (Chemin de génération Java).

Erreurs et avertissements propres au service

Nous vous recommandons de toujours laisser la hiérarchie de l'enregistreur d'événements définie avec la valeur « WARN » pour intercepter les messages importants des bibliothèques clientes. Par exemple, si le Amazon S3 client détecte que votre application n'a pas correctement fermé une application `InputStream` et qu'elle est susceptible de provoquer une fuite de ressources, le client S3 le signale par le biais d'un message d'avertissement envoyé aux journaux. Il est ainsi possible de s'assurer que les messages sont enregistrés au cas où le client rencontrerait des problèmes de gestion des demandes ou des réponses.

Le fichier `log4j.properties` suivant définit `rootLogger` avec la valeur `WARN`, ce qui entraîne l'inclusion des messages d'avertissement et d'erreur de tous les enregistreurs d'événements de la hiérarchie « `com.amazonaws` ». Vous pouvez aussi définir explicitement l'enregistreur d'événements `com.amazonaws` avec la valeur `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
```

Journalisation récapitulative des demandes et des réponses

Chaque demande envoyée à un Service AWS génère un identifiant de AWS demande unique qui est utile si vous rencontrez un problème avec le traitement d'une demande par un Service AWS . AWS Les identifiants de demande sont accessibles par programmation via les objets `Exception` du SDK en cas d'échec d'un appel de service, et peuvent également être signalés via le niveau de journalisation `DEBUG` dans l'enregistreur « `com.amazonaws.request` ».

Le fichier `log4j.properties` suivant permet de résumer les demandes et les réponses, y compris AWS les identifiants des demandes.

```
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
```

Voici un exemple de la sortie du journal.

```
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-00000000000000, AWSAccessKeyId: ACCESSKEYID,
Version: 2009-10-16, SignatureVersion: 2, Timestamp:
2009-12-17T17:53:04.467Z, Signature:
9WcgfPwTobvLVcphybrdN7P713uH0oviYQ4yZ+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
```

Journalisation du réseau filaire détaillée

Dans certains cas, il peut être utile de voir les demandes et réponses exactes qu'ils AWS SDK for Java envoient et reçoivent. Vous ne devez pas activer cette journalisation dans les systèmes de production, car l'écriture de demandes volumineuses (par exemple, le téléchargement d'un fichier Amazon S3) ou de réponses peut ralentir considérablement une application. Si vous avez vraiment besoin d'accéder à ces informations, vous pouvez les activer temporairement via l'enregistreur Apache HttpClient 4. L'activation du niveau DEBUG sur l'enregistreur d'événements

`org.apache.http.wire` permet la journalisation de toutes les données de demande et de réponse.

Le fichier `log4j.properties` suivant active la journalisation complète dans Apache HttpClient 4 et ne doit être activé que temporairement car cela peut avoir un impact significatif sur les performances de votre application.

```
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
```

Journalisation des métriques de latence

L'enregistreur d'événements de latence peut s'avérer utile si vous voulez résoudre des problèmes et que vous souhaitez voir des métriques permettant entre autres de déterminer quel processus prend le plus de temps, ou si le côté serveur ou client a la plus grande latence. Définissez l'enregistreur d'événements `com.amazonaws.latency` sur `DEBUG` pour activer cet enregistreur d'événements.

Note

Cet enregistreur d'événements est disponible uniquement si les métriques SDK sont activées. Pour en savoir plus sur le package de métriques du SDK, consultez [Enabling Metrics for the 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
```

Voici un exemple de la sortie du journal.

```
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]
```

Configuration de client

AWS SDK for Java Cela vous permet de modifier la configuration par défaut du client, ce qui est utile lorsque vous souhaitez :

- Se connecter à Internet via un proxy
- Modifier les paramètres de transport HTTP, tels que le délai de connexion et les nouvelles tentatives de demande
- Spécifier des conseils sur la taille de la mémoire tampon du socket TCP

Configuration de proxy

Lorsque vous créez un objet client, vous pouvez transmettre un [ClientConfiguration](#) objet facultatif pour personnaliser la configuration du client.

Si vous vous connectez à Internet via un serveur proxy, vous devez configurer les paramètres du serveur proxy (hôte proxy, port et nom d'utilisateur/mot de passe) via l'objet `ClientConfiguration`.

Configuration du transport HTTP

Vous pouvez configurer plusieurs options de transport HTTP à l'aide de l'[ClientConfiguration](#) objet. De nouvelles options sont parfois ajoutées ; pour voir la liste complète des options que vous pouvez récupérer ou définir, consultez la référence de l' AWS SDK for Java API.

Note

Chacune des valeurs configurables possède une valeur par défaut définie par une constante. Pour obtenir la liste des valeurs constantes pour `ClientConfiguration`, consultez la section [Valeurs de champ constantes](#) dans la référence de l' AWS SDK for Java API.

Connexions maximales

Vous pouvez définir le nombre maximum autorisé de connexions HTTP ouvertes à l'aide du [ClientConfiguration.setMaxConnections](#) méthode.

Important

Définissez le nombre maximal de connexions de telle sorte qu'il corresponde au nombre de transactions simultanées. Vous éviterez ainsi des contentions de connexions et une dégradation des performances. Pour connaître la valeur maximale de connexions par défaut, consultez la section [Valeurs de champ constantes](#) dans la référence de l' AWS SDK for Java API.

Délais et gestion des erreurs

Vous pouvez définir des options liées aux délais et à la gestion des erreurs avec les connexions HTTP.

- Délai de connexion

Le délai de connexion correspond à la durée (en millisecondes) pendant laquelle la connexion HTTP attend pour établir une connexion avant d'abandonner. La valeur par défaut est 10 000 ms.

Pour définir vous-même cette valeur, utilisez le [ClientConfiguration.setConnectionTimeout](#) méthode.

- Durée de vie (TTL) de la connexion

Par défaut, le kit SDK tente de réutiliser les connexions HTTP aussi longtemps que possible. Dans les situations d'échec où une connexion est établie vers un serveur qui a été mis hors service, le fait d'avoir une durée de vie finie peut aider à la récupération de l'application. Par exemple, la définition d'une durée de vie de 15 minutes garantit que, même si vous avez une connexion établie

avec un serveur qui rencontre des problèmes, vous rétablirez une connexion à un nouveau serveur dans un délai de 15 minutes.

Pour définir le TTL de connexion HTTP, utilisez la méthode [ClientConfiguration.setConnectionTTL](#).

- Nombre maximal de tentatives en cas d'erreur

Par défaut, le nombre maximal de tentatives en cas d'erreur est de 3. Vous pouvez définir une valeur différente en utilisant le [ClientConfiguration.setMaxErrorRéessayer](#) la méthode.

Adresse locale

Pour définir l'adresse locale à laquelle le client HTTP doit se lier, utilisez [ClientConfiguration.setLocalAddress](#).

Conseils sur la taille de la mémoire tampon du socket TCP

Les utilisateurs expérimentés qui souhaitent ajuster les paramètres TCP de bas niveau peuvent également définir des indications sur la taille de la mémoire tampon TCP via l'objet [ClientConfiguration](#). La majorité des utilisateurs n'aura jamais besoin de modifier ces valeurs, mais elles sont fournies pour les utilisateurs avancés.

Les tailles optimales de mémoire tampon TCP d'une application dépendent fortement de la configuration et des capacités du réseau et du système d'exploitation. Par exemple, la plupart des systèmes d'exploitation modernes fournissent une logique de réglage automatique pour les tailles de mémoire tampon de socket TCP. Il peut en résulter un impact important sur les performances des connexions TCP qui sont maintenues ouvertes assez longtemps pour que le réglage automatique optimise les tailles de mémoire tampon.

Les tailles de mémoire tampon élevées (par exemple, 2 Mo) permettent au système d'exploitation de placer en mémoire tampon plus de données sans avoir besoin que le serveur distant accuse réception des informations et peuvent ainsi se révéler particulièrement utiles quand le réseau présente une latence élevée.

Il s'agit uniquement d'un conseil ; il se peut que le système d'exploitation ne le suive pas. Lors de l'utilisation de cette option, les utilisateurs doivent toujours vérifier les valeurs par défaut et les limites configurées du système d'exploitation. La plupart des systèmes d'exploitation ont une taille maximale de mémoire tampon TCP et ne vous permettent pas de dépasser ce seuil, sauf si vous augmentez explicitement la taille maximale de la mémoire tampon TCP.

De nombreuses ressources sont disponibles pour vous aider à configurer les tailles de mémoire tampon TCP et les paramètres TCP spécifiques au système d'exploitation, y compris les éléments suivants :

- [Réglage de l'hôte](#)

Stratégies de contrôle d'accès

AWS les politiques de contrôle d'accès vous permettent de définir des contrôles d'accès précis sur vos AWS ressources. Une stratégie de contrôle d'accès se compose d'un ensemble de déclarations qui se présentent sous la forme suivante :

Le compte A est autorisé à exécuter l'action B sur la ressource C lorsque la condition D s'applique.

Où :

- A est le principal : celui Compte AWS qui fait une demande pour accéder à l'une de vos AWS ressources ou pour la modifier.
- B est l'action : mode d'accès ou de modification de votre AWS ressource, par exemple en envoyant un message à une Amazon SQS file d'attente ou en stockant un objet dans un Amazon S3 compartiment.
- C est la ressource : AWS entité à laquelle le principal souhaite accéder, telle qu'une Amazon SQS file d'attente ou un objet stocké dans celui-ci Amazon S3.
- D est un ensemble de conditions : les contraintes facultatives qui spécifient quand autoriser ou refuser l'accès au principal pour accéder à votre ressource. De nombreuses conditions expressives sont disponibles, certaines spécifiques à chaque service. Par exemple, vous pouvez utiliser des conditions de date pour autoriser l'accès à vos ressources uniquement après ou avant un moment spécifique.

Amazon S3 Exemple

L'exemple suivant illustre une politique qui permet à quiconque d'accéder à tous les objets d'un compartiment, mais restreint l'accès au téléchargement d'objets vers ce compartiment à deux options spécifiques Compte AWS(en plus du compte du propriétaire du compartiment).

```
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 Exemple

L'une des utilisations courantes des politiques consiste à autoriser une Amazon SQS file d'attente à recevoir des messages provenant d'une rubrique 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));
```

Exemple Amazon SNS

Certains services proposent des conditions supplémentaires qui peuvent être utilisées dans les politiques. Amazon SNS fournit les conditions permettant d'autoriser ou de refuser les abonnements aux rubriques SNS en fonction du protocole (e-mail, HTTP, HTTPS, etc. Amazon SQS) et du point de terminaison (adresse e-mail, URL, Amazon SQS ARN) de la demande d'abonnement à une rubrique.

```
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()));
```

Définissez le TTL de la JVM pour les recherches de noms DNS

La machine virtuelle Java (JVM) met en cache les recherches de nom DNS. Lorsque la JVM convertit un nom d'hôte en adresse IP, elle met l'adresse IP en cache pendant une période spécifiée, connue sous le nom de time-to-live(TTL).

Étant donné que les AWS ressources utilisent des entrées de nom DNS qui changent occasionnellement, nous vous recommandons de configurer votre JVM avec une valeur TTL de 5 secondes. Ainsi, lorsque l'adresse IP d'une ressource change, votre application peut recevoir et utiliser la nouvelle adresse IP de la ressource en interrogeant le DNS.

Dans certaines configurations Java, la durée de vie par défaut de la JVM est définie de façon à ce que la JVM n'actualise jamais les entrées DNS tant qu'elle n'est pas redémarrée. Ainsi, si l'adresse IP d'une AWS ressource change alors que votre application est toujours en cours d'exécution, elle ne pourra pas utiliser cette ressource tant que vous n'aurez pas redémarré manuellement la JVM et que les informations IP mises en cache ne seront pas actualisées. Dans ce cas, il est essentiel de définir la durée de vie de la JVM de façon à ce que ses informations IP mises en cache soient régulièrement actualisées.

Comment configurer le JVM TTL

Pour modifier le TTL de la JVM, définissez la valeur de la propriété de sécurité [networkaddress.cache.ttl](#), définissez la propriété dans le `networkaddress.cache.ttl` `$JAVA_HOME/jre/lib/security/java.security` fichier pour Java 8 ou dans le fichier pour Java 11 ou supérieur. `$JAVA_HOME/conf/security/java.security`

Ce qui suit est un extrait d'un `java.security` fichier qui montre que le cache TTL est réglé sur 5 secondes.

```
#
# 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
...
```

Toutes les applications qui s'exécutent sur la JVM représentée par la variable d'\$JAVA_HOME environnement utilisent ce paramètre.

Activation des métriques pour AWS SDK for Java

Ils AWS SDK for Java peuvent générer des métriques à des fins de visualisation et de surveillance avec [Amazon CloudWatch](#) qui mesurent :

- les performances de votre application lors de l'accès AWS
- les performances de vos JVM lorsqu'elles sont utilisées avec AWS
- des détails sur l'environnement d'exécution comme la mémoire de segment, le nombre de threads et les descripteurs de fichier ouverts

Comment activer la génération de métriques du SDK Java

Vous devez ajouter la dépendance Maven suivante pour permettre au SDK d'envoyer des métriques à. 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>
```

* Remplacez le numéro de version par la dernière version du SDK disponible sur [Maven Central](#).

AWS SDK for Java les métriques sont désactivées par défaut. Pour l'activer dans votre environnement de développement local, incluez une propriété système pointant vers votre fichier d'informations AWS de sécurité lors du démarrage de la JVM. Par exemple :

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

Vous devez spécifier le chemin d'accès à votre fichier d'identification afin que le SDK puisse télécharger les points de données collectés pour CloudWatch une analyse ultérieure.

Note

Si vous accédez AWS depuis une Amazon EC2 instance à l'aide du service de métadonnées d' Amazon EC2 instance, vous n'avez pas besoin de spécifier de fichier d'informations d'identification. Dans ce cas, vous devez seulement spécifier :

```
-Dcom.amazonaws.sdk.enableDefaultMetrics
```

Toutes les métriques capturées par le AWS SDK for Java se trouvent sous l'espace de noms AWSSDK/Java et sont téléchargées dans la région CloudWatch par défaut (us-east-1). Pour changer de région, spécifiez votre région en utilisant l'attribut `cloudwatchRegion` dans la propriété système. Par exemple, pour définir la CloudWatch région sur us-east-1, utilisez :

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

Une fois la fonctionnalité activée, chaque fois qu'une demande de service est envoyée, des points AWS de données métriques sont générés AWS SDK for Java, mis en file d'attente pour un résumé

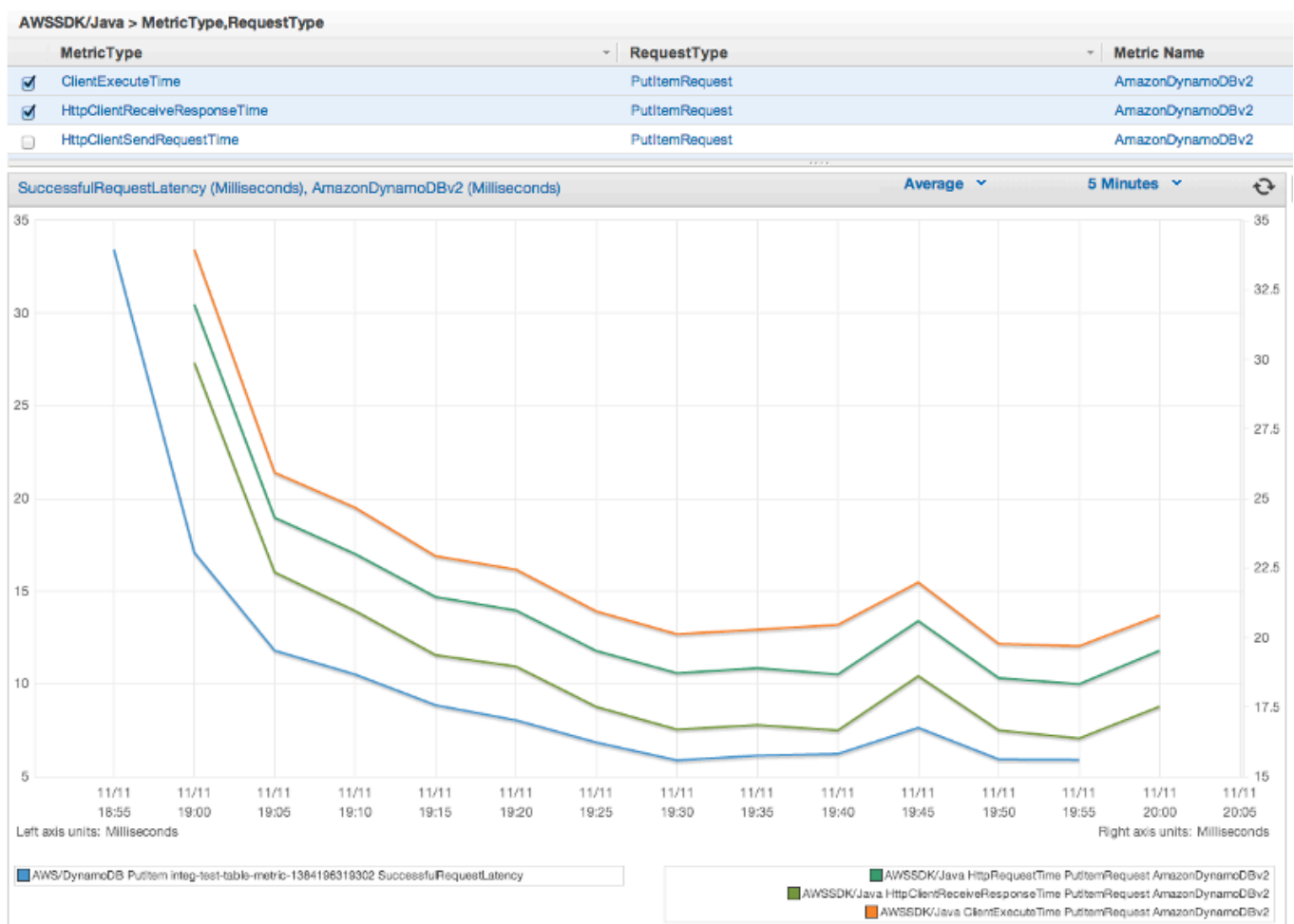
statistique et téléchargés de manière asynchrone CloudWatch environ une fois par minute. Une fois les métriques téléchargées, vous pouvez les visualiser à l'aide de [AWS Management Console](#) et définir des alarmes en cas de problèmes potentiels tels que les fuites de mémoire, les fuites de descripteurs de fichiers, etc.

Types de métrique disponibles

L'ensemble de métriques par défaut est divisé en trois catégories principales :

AWS Métriques des demandes

- Couvrent des domaines tels que la latence de la demande/réponse HTTP, le nombre de demandes, les exceptions et les nouvelles tentatives.



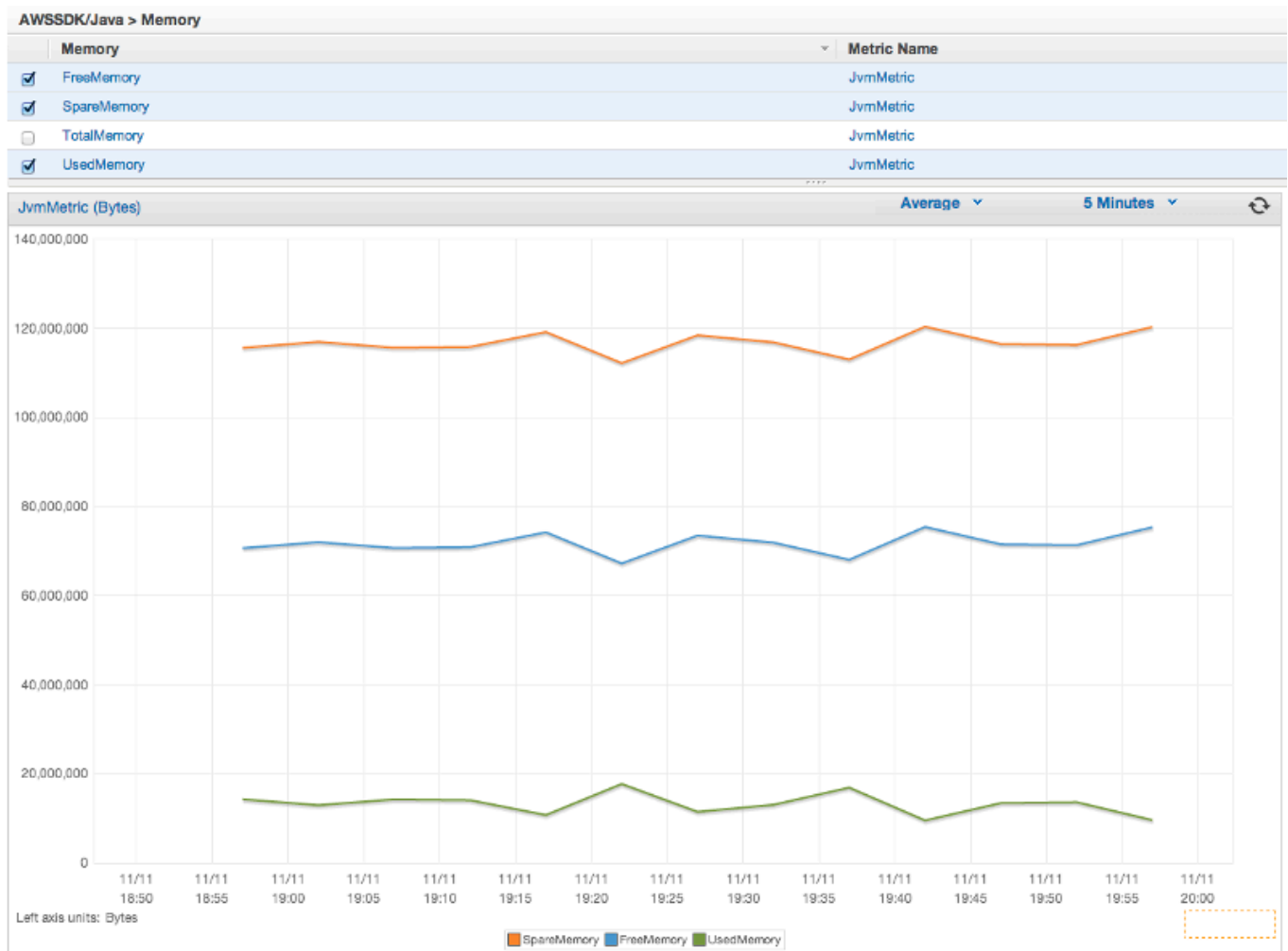
Service AWS Métriques

- Incluez Service AWS des données spécifiques, telles que le débit et le nombre d'octets pour les chargements et téléchargements S3.



Métriques machine

- Couvrent l'environnement d'exécution, y compris la mémoire de segment, le nombre de threads et les descripteurs de fichier ouverts.



Si vous souhaitez exclure les métriques machine, ajoutez `excludeMachineMetrics` à la propriété système :

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

En savoir plus

- Consultez le [récapitulatif du package amazonaws/metrics](#) pour voir la liste complète des types de métriques de base prédéfinies.
- Découvrez comment CloudWatch utiliser le AWS SDK for Java dans [CloudWatch Exemples d'utilisation du AWS SDK for Java](#).

- Pour en savoir plus sur le réglage des performances, consultez [le billet de blog Tuning the AWS SDK for Java to Improve Resiliency](#).

Exemples de code de l'AWS SDK for Java

Cette section fournit des didacticiels et des exemples d'utilisation de laAWS SDK for Java version 1 pour programmerAWS des services.

Trouvez le code source de ces exemples et d'autres dans le [référentiel d'exemples de code deAWS documentation surGitHub](#).

Pour proposer un nouvel exemple de code que l'équipe de documentation AWS pourrait envisager de produire, créez une nouvelle demande. L'équipe cherche à produire des exemples de code qui couvrent des scénarios et des cas d'utilisation plus larges, plutôt que de simples extraits de code qui couvrent uniquement les appels d'API individuels. Pour obtenir des instructions, consultez les [directives de contribution](#) dans le référentiel d'exemples de code surGitHub..

AWS SDK for Java2.x

En 2018,AWS a publié le [AWS SDK for Java 2.x](#). Ce guide contient des instructions sur l'utilisation de la dernière version du SDK Java ainsi que des exemples de code.

Note

Consultez [Documentation et ressources supplémentaires](#) pour obtenir plus d'exemples et les ressources supplémentaires disponibles pour les développeurs AWS SDK for Java !

Exemples CloudWatch utilisant leAWS SDK for Java

Cette section fournit des exemples de programmation d'[CloudWatch](#) à l'aide du kit [AWS SDK for Java](#).

Amazon CloudWatch surveille vosAmazon Web Services(AWS) et les applications que vous exécutez surAWSen temps réel Vous pouvez utiliser CloudWatch pour recueillir et suivre les métriques, qui sont des variables que vous pouvez mesurer pour vos ressources et applications. CloudWatch Les alarmes envoient des notifications ou apportent des modifications automatiquement aux ressources surveillées, selon les règles que vous définissez.

Pour en savoir plus sur CloudWatch, consultez le [Guide de l'utilisateurAmazon CloudWatch](#).

Note

Les exemples incluent uniquement le code nécessaire pour démontrer chaque technique. L'exemple de code complet [est disponible sur GitHub](#). À partir de là, vous pouvez télécharger un fichier source unique ou cloner le référentiel en local pour obtenir tous les exemples à générer et exécuter.

Rubriques

- [Obtention de métriques à partir de CloudWatch](#)
- [Publication de données de métriques personnalisées](#)
- [Utilisation d' CloudWatch Alarmes](#)
- [Utilisation d'actions d'alarme dans CloudWatch](#)
- [Envoi d'événements à CloudWatch](#)

Obtention de métriques à partir de CloudWatch

Affichage de la liste des métriques

Pour lister CloudWatch métriques, créez un [ListMetricsRequest](#) et appelez le client `AmazonCloudWatchClient.listMetrics` méthode. Vous pouvez utiliser `ListMetricsRequest` pour filtrer les métriques renvoyées par espace de noms, nom de métrique ou dimension.

Note

Liste des mesures et des dimensions publiées par AWS les services peuvent être trouvés dans le [https://docs-aws-amazon-com-AmazonCloudWatch-Latest-Monitoring-CW-Support-for-AWS-html](https://docs.aws.amazon.com/AmazonCloudWatch-Latest-Monitoring-CW-Support-for-AWS-html) [Amazon CloudWatch Référence de dimensions et de métriques dans le Amazon CloudWatch Guide de l'utilisateur .

Importations

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

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

Code

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

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

boolean done = false;

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

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

    request.setNextToken(response.getNextToken());

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

Les métriques sont renvoyées dans un objet [ListMetricsResult](#) en appelant la méthode `getMetrics` de cet objet. Les résultats peuvent être paginés. Pour récupérer le lot suivant de résultats, appelez `setNextToken` sur l'objet de demande d'origine avec la valeur de retour de la méthode `getNextToken` de l'objet `ListMetricsResult`, et retransmettez l'objet de demande modifié vers un autre appel de `listMetrics`.

En savoir plus

- [ListMetrics](#) dans le Amazon CloudWatch Référence d'API.

Publication de données de métriques personnalisées

Un certain nombre d'AWS publication des services [leurs propres mesures](#) dans les espaces de noms commençant par »AWS « Vous pouvez également publier des données de métriques personnalisées en utilisant votre propre espace de noms (dans la mesure où il ne commence pas par »AWS «).

Publication de données de métriques personnalisées

Pour publier vos propres données de métriques, appelez `putMetricData` avec un [PutMetricDataRequest](#). L'objet `PutMetricDataRequest` doit inclure l'espace de noms personnalisé à utiliser pour les données et des informations sur le point de données proprement dit dans un objet [MetricDatum](#).

Note

Vous ne pouvez pas spécifier un espace de noms commençant par »AWS «. Espaces de noms commençant par »AWS « sont réservés pour un usage par Amazon Web Services Produits .

Importations

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

Code

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

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

MetricDatum datum = new MetricDatum()
```

```
.withMetricName("PAGES_VISITED")
.withUnit(StandardUnit.None)
.withValue(data_point)
.withDimensions(dimension);

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

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

En savoir plus

- [A l'aide de Amazon CloudWatch Métriques](#) dans le Amazon CloudWatch Guide de l'utilisateur .
- [AWS Espaces de noms](#) dans le Amazon CloudWatch Guide de l'utilisateur .
- [PutMetricData](#) dans le Amazon CloudWatch Référence API.

Utilisation d' CloudWatch Alarmes

Créer une alarme

Pour créer une alarme basée sur un CloudWatch , appelez le client `AmazonCloudWatchClient` `putMetricAlarm` méthode avec un [PutMetricAlarmRequest](#) remplie des conditions d'alarme.

Importations

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.ComparisonOperator;
import com.amazonaws.services.cloudwatch.model.Dimension;
import com.amazonaws.services.cloudwatch.model.PutMetricAlarmRequest;
import com.amazonaws.services.cloudwatch.model.PutMetricAlarmResult;
import com.amazonaws.services.cloudwatch.model.StandardUnit;
import com.amazonaws.services.cloudwatch.model.Statistic;
```

Code

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

```
Dimension dimension = new Dimension()
    .withName("InstanceId")
    .withValue(instanceId);

PutMetricAlarmRequest request = new PutMetricAlarmRequest()
    .withAlarmName(alarmName)
    .withComparisonOperator(
        ComparisonOperator.GreaterThanThreshold)
    .withEvaluationPeriods(1)
    .withMetricName("CPUUtilization")
    .withNamespace("{AWS}/EC2")
    .withPeriod(60)
    .withStatistic(Statistic.Average)
    .withThreshold(70.0)
    .withActionsEnabled(false)
    .withAlarmDescription(
        "Alarm when server CPU utilization exceeds 70%")
    .withUnit(StandardUnit.Seconds)
    .withDimensions(dimension);

PutMetricAlarmResult response = cw.putMetricAlarm(request);
```

Affichage des alarmes

Pour afficher le CloudWatch Alarmes que vous avez créées, appelez `describeAlarms` méthode avec un [DescribeAlarmsRequest](#) que vous pouvez utiliser pour définir des options pour le résultat.

Importations

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.DescribeAlarmsRequest;
import com.amazonaws.services.cloudwatch.model.DescribeAlarmsResult;
import com.amazonaws.services.cloudwatch.model.MetricAlarm;
```

Code

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

boolean done = false;
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
```

```
while(!done) {  
  
    DescribeAlarmsResult response = cw.describeAlarms(request);  
  
    for(MetricAlarm alarm : response.getMetricAlarms()) {  
        System.out.printf("Retrieved alarm %s", alarm.getAlarmName());  
    }  
  
    request.setNextToken(response.getNextToken());  
  
    if(response.getNextToken() == null) {  
        done = true;  
    }  
}
```

La liste des alarmes peut être obtenue en appelant `getMetricAlarms` sur l'objet [DescribeAlarmsResult](#) renvoyé par `describeAlarms`.

Les résultats peuvent être paginés. Pour récupérer le lot suivant de résultats, appelez `setNextToken` sur l'objet de demande d'origine avec la valeur de retour de la méthode `getNextToken` de l'objet `DescribeAlarmsResult`, et retransmettez l'objet de demande modifié vers un autre appel de `describeAlarms`.

Note

Vous pouvez également récupérer des alarmes pour une métrique spécifique à l'aide de `describeAlarmsForMetric` méthode. Son utilisation est similaire à `describeAlarms`.

Suppression d'alarmes

Pour supprimer CloudWatch alarmes, appelez le client `AmazonCloudWatchdeleteAlarms` méthode avec un [DeleteAlarmsRequest](#) contenant un ou plusieurs noms d'alarmes que vous voulez supprimer.

Importations

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


Code

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

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

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

En savoir plus

- [Création Amazon CloudWatch Alarmes](#) dans le Amazon CloudWatch Guide de l'utilisateur
- [PutMetricAlarm](#) dans le Amazon CloudWatch API Reference
- [DescribeAlarms](#) dans le Amazon CloudWatch API Reference
- [DeleteAlarms](#) dans le Amazon CloudWatch API Reference

Utilisation d'actions d'alarme dans CloudWatch

A l'aide de CloudWatch d'actions d'alarme, vous pouvez créer des alarmes qui exécutent des actions tels que l'arrêt, la résiliation, le redémarrage ou la récupération automatiques Amazon EC2 instances.

Note

Des actions d'alarme peuvent être ajoutées à une alarme en utilisant la méthode [de PutMetricAlarmRequest.setAlarmActions](#) lors de la [création d'une alarme](#).

Activation d'actions d'alarme

Pour activer les actions d'alarme pour un CloudWatch , appelez le client `AmazonCloudWatch` avec un [EnableAlarmActionsRequest](#) contenant un ou plusieurs noms d'alarmes dont vous souhaitez activer les actions.

Importations

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

```
import com.amazonaws.services.cloudwatch.model.EnableAlarmActionsRequest;
import com.amazonaws.services.cloudwatch.model.EnableAlarmActionsResult;
```

Code

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

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

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

Désactivation d'actions d'alarme

Pour désactiver les actions d'alarme pour un CloudWatch , appelez le client `AmazonCloudWatch` avec un `DisableAlarmActionsRequest` contenant un ou plusieurs noms d'alarmes dont vous souhaitez désactiver les actions.

Importations

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.DisableAlarmActionsRequest;
import com.amazonaws.services.cloudwatch.model.DisableAlarmActionsResult;
```

Code

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

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

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

En savoir plus

- [Création d'alarmes qui arrêtent, mettent hors service, redémarrent ou récupèrent une instance](#) dans le `Amazon CloudWatch` Guide de l'utilisateur

- [PutMetricAlarm](#) dans le Amazon CloudWatch API Reference
- [EnableAlarmActions](#) dans le Amazon CloudWatch API Reference
- [DisableAlarmActions](#) dans le Amazon CloudWatch API Reference

Envoi d'événements à CloudWatch

CloudWatchEvents fournit un flux d'événements système en temps quasi réel qui décrivent les modifications apportées à AWS ressources pour Amazon EC2 instances, Lambda Fonctions Kinesis flux, Amazon EC2 tâches, Step Functions machines d'état, Amazon SNS sujets, Amazon SQS files d'attente ou cibles intégrées. À l'aide de règles simples, vous pouvez faire correspondre les événements et les acheminer vers un ou plusieurs flux ou fonctions cibles.

Ajout d'événements

Pour ajouter des informations personnalisées CloudWatch événements, appelez le client Amazon CloudWatch Events `Client.putEvent` avec une méthode [PutEventsRequest](#) qui contient un ou plusieurs objets [PutEventsRequestEntry](#) objets qui fournissent des détails sur chaque événement. Vous pouvez spécifier plusieurs paramètres pour l'entrée, tels que la source et le type de l'événement, les ressources associées à l'événement, et ainsi de suite.

Note

Vous pouvez spécifier un maximum de 10 événements par appel de `putEvents`.

Importations

```
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutEventsRequest;
import com.amazonaws.services.cloudwatchevents.model.PutEventsRequestEntry;
import com.amazonaws.services.cloudwatchevents.model.PutEventsResult;
```

Code

```
final AmazonCloudWatchEvents cwe =
```

```
AmazonCloudWatchEventsClientBuilder.defaultClient();

final String EVENT_DETAILS =
    "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

PutEventsRequestEntry request_entry = new PutEventsRequestEntry()
    .withDetail(EVENT_DETAILS)
    .withDetailType("sampleSubmitted")
    .withResources(resource_arn)
    .withSource("aws-sdk-java-cloudwatch-example");

PutEventsRequest request = new PutEventsRequest()
    .withEntries(request_entry);

PutEventsResult response = cwe.putEvents(request);
```

Ajout de règles

Pour créer ou mettre à jour une règle, appelez `putRule` avec une méthode [PutRuleRequest](#) [PUR](#) avec le nom de la règle et des paramètres facultatifs tels que le [Modèle d'événement](#), IAM rôle à associer à la règle, et un [Expression de planification](#) qui décrit la fréquence d'exécution de la règle.

Importations

```
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutRuleRequest;
import com.amazonaws.services.cloudwatchevents.model.PutRuleResult;
import com.amazonaws.services.cloudwatchevents.model.RuleState;
```

Code

```
final AmazonCloudWatchEvents cwe =
    AmazonCloudWatchEventsClientBuilder.defaultClient();

PutRuleRequest request = new PutRuleRequest()
    .withName(rule_name)
    .withRoleArn(role_arn)
    .withScheduleExpression("rate(5 minutes)")
    .withState(RuleState.ENABLED);

PutRuleResult response = cwe.putRule(request);
```

Ajout de cibles

Les cibles sont les ressources appelées lorsqu'une règle est déclenchée. Parmi les exemples de cibles, citons les instances Amazon EC2, les fonctions Lambda, les flux Kinesis, les tâches Amazon ECS, les machines d'état Step Functions et les cibles intégrées.

Pour ajouter une cible à une règle, appelez `putTargets` avec une méthode [PutTargetsRequest](#) contenant la règle à mettre à jour et la liste des cibles à ajouter à la règle.

Importations

```
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEvents;
import com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClientBuilder;
import com.amazonaws.services.cloudwatchevents.model.PutTargetsRequest;
import com.amazonaws.services.cloudwatchevents.model.PutTargetsResult;
import com.amazonaws.services.cloudwatchevents.model.Target;
```

Code

```
final AmazonCloudWatchEvents cwe =
    AmazonCloudWatchEventsClientBuilder.defaultClient();

Target target = new Target()
    .withArn(function_arn)
    .withId(target_id);

PutTargetsRequest request = new PutTargetsRequest()
    .withTargets(target)
    .withRule(rule_name);

PutTargetsResult response = cwe.putTargets(request);
```

En savoir plus

- [Ajout d'événements avec PutEvents](#) dans le Amazon CloudWatch Events Guide de l'utilisateur
- [Expression de planification des règles](#) dans le Amazon CloudWatch Events Guide de l'utilisateur
- [Types d'événements pour CloudWatchEvents \(Événements\)](#) dans le Amazon CloudWatch Events Guide de l'utilisateur
- [Événements et modèles d'événements](#) dans le Amazon CloudWatch Events Guide de l'utilisateur

- [PutEvents](#) dans le Amazon CloudWatch Events API Reference
- [PutTargets](#) dans le Amazon CloudWatch Events API Reference
- [PutRule](#) dans le Amazon CloudWatch Events API Reference

DynamoDB Exemples avec utilisation de l’AWS SDK for Java

Cette section fournit des exemples de programmation d'[DynamoDB](#) à l'aide du kit [AWS SDK for Java](#).

Note

Les exemples incluent uniquement le code nécessaire pour démontrer chaque technique. L'exemple de code complet [est disponible sur GitHub](#). À partir de là, vous pouvez télécharger un fichier source unique ou cloner le référentiel en local pour obtenir tous les exemples à générer et exécuter.

Rubriques

- [Utilisation de tables dans DynamoDB](#)
- [Utilisation des éléments dans DynamoDB](#)

Utilisation de tables dans DynamoDB

Les tables sont les conteneurs pour tous les éléments d'une base de données DynamoDB. Pour pouvoir ajouter ou supprimer des données dans DynamoDB, vous devez créer une table.

Pour chaque table, vous devez définir :

- Un nom de table unique pour le compte et la région.
- Une clé primaire pour laquelle chaque valeur doit être unique : deux éléments de votre table ne peuvent pas avoir la même valeur de clé primaire.

Une clé primaire peut être simple, constituée d'une seule clé de partition (HASH) ou composite, constituée d'une partition et d'une clé de tri (RANGE).

Chaque valeur clé est associée à un type de données, énuméré par la [ScalarAttributeType](#) classe. La valeur de la clé peut être binaire (B), numérique (N) ou de type chaîne (S). Pour plus

d'informations, consultez la section [Règles de dénomination et types de données](#) dans le Guide duAmazon DynamoDB développeur.

- Des valeurs de débit alloué qui définissent le nombre d'unités de capacité en lecture/écriture réservées pour la table.

Note

Amazon DynamoDBla [tarification](#) est basée sur les valeurs de débit provisionnées que vous définissez pour vos tables. Ne réservez donc que la capacité dont vous pensez avoir besoin pour votre table.

Le débit alloué pour une table peut être modifié à tout moment pour que vous puissiez ajuster la capacité si vos besoins évoluent.

Création d'une table

Utilisez la `createTable` méthode du [DynamoDBclient](#) pour créer une nouvelleDynamoDB table. Vous devez créer des attributs de table et un schéma de table qui sont utilisés pour identifier la clé primaire de votre table. Vous devez également fournir des valeurs initiales de débit alloué et un nom de table. Définissez uniquement des attributs de table clé lors de la création de votre table DynamoDB.

Note

Si une table portant le nom que vous avez choisi existe déjà, une table [AmazonServiceException](#)est générée.

Importations

```
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;
```

Création d'une table avec une clé primaire simple

Ce code crée une table avec une clé primaire simple ("Name").

Code

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

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

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

Voir l'[exemple complet](#) sur GitHub.

Création d'une table avec une clé primaire composite

Ajoutez un autre [AttributeDefinition](#) et [KeySchemaElement](#) à [CreateTableRequest](#).

Code

```
CreateTableRequest request = new CreateTableRequest()
    .withAttributeDefinitions(
        new AttributeDefinition("Language", ScalarAttributeType.S),
        new AttributeDefinition("Greeting", ScalarAttributeType.S))
    .withKeySchema(
        new KeySchemaElement("Language", KeyType.HASH),
        new KeySchemaElement("Greeting", KeyType.RANGE))
    .withProvisionedThroughput(
```



```
new ProvisionedThroughput(new Long(10), new Long(10)))
.withTableName(table_name);
```

Voir l'[exemple complet](#) sur GitHub.

Affichage d'une liste de tables

Vous pouvez répertorier les tables d'une région particulière en appelant la `listTables` méthode du [DynamoDBClient](#).

Note

Si la table nommée n'existe pas pour votre compte et votre région, un [ResourceNotFoundException](#) est généré.

Importations

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

Code

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

ListTablesRequest request;

boolean more_tables = true;
String last_name = null;

while(more_tables) {
    try {
        if (last_name == null) {
            request = new ListTablesRequest().withLimit(10);
        }
        else {
            request = new ListTablesRequest()
                .withLimit(10)
                .withExclusiveStartTableName(last_name);
        }
    }
}
```

```
    }

    ListTablesResult table_list = ddb.listTables(request);
    List<String> table_names = table_list.getTableNames();

    if (table_names.size() > 0) {
        for (String cur_name : table_names) {
            System.out.format("* %s\n", cur_name);
        }
    } else {
        System.out.println("No tables found!");
        System.exit(0);
    }

    last_name = table_list.getLastEvaluatedTableName();
    if (last_name == null) {
        more_tables = false;
    }
}
```

Par défaut, jusqu'à 100 tables sont renvoyées par appel. Utilisez `getLastEvaluatedTableName` sur l'[ListTablesResult](#) objet renvoyé pour obtenir la dernière table évaluée. Vous pouvez utiliser cette valeur pour démarrer la liste après la dernière valeur renvoyée de la liste précédente.

Voir l'[exemple complet](#) sur GitHub.

Description d'une table (obtention d'informations sur celle-ci)

Appelez la `describeTable` méthode du [DynamoDBClient](#).

Note

Si la table nommée n'existe pas pour votre compte et votre région, un [ResourceNotFoundException](#) est généré.

Importations

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

```
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughputDescription;
import com.amazonaws.services.dynamodbv2.model.TableDescription;
```

Code

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

try {
    TableDescription table_info =
        ddb.describeTable(table_name).getTable();

    if (table_info != null) {
        System.out.format("Table name   : %s\n",
            table_info.getTableName());
        System.out.format("Table ARN   : %s\n",
            table_info.getTableArn());
        System.out.format("Status      : %s\n",
            table_info.getTableStatus());
        System.out.format("Item count  : %d\n",
            table_info.getItemCount().longValue());
        System.out.format("Size (bytes): %d\n",
            table_info.getTableSizeBytes().longValue());

        ProvisionedThroughputDescription throughput_info =
            table_info.getProvisionedThroughput();
        System.out.println("Throughput");
        System.out.format("  Read Capacity : %d\n",
            throughput_info.getReadCapacityUnits().longValue());
        System.out.format("  Write Capacity: %d\n",
            throughput_info.getWriteCapacityUnits().longValue());

        List<AttributeDefinition> attributes =
            table_info.getAttributeDefinitions();
        System.out.println("Attributes");
        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n",
                a.getAttributeName(), a.getAttributeType());
        }
    }
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Voir l'[exemple complet](#) surGitHub.

Modification (mise à jour) d'une table

Vous pouvez modifier les valeurs de débit provisionnées de votre table à tout moment en appelant la `updateTable` méthode du [DynamoDBClient](#).

Note

Si la table nommée n'existe pas pour votre compte et votre région, un [ResourceNotFoundException](#) est généré.

Importations

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

Code

```
ProvisionedThroughput table_throughput = new ProvisionedThroughput(
    read_capacity, write_capacity);

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    ddb.updateTable(table_name, table_throughput);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Voir l'[exemple complet](#) surGitHub.

Suppression d'une table

Appellez la `deleteTable` méthode du [DynamoDBClient](#) et transmettez-lui le nom de la table.

Note

Si la table nommée n'existe pas pour votre compte et votre région, un [ResourceNotFoundException](#) est généré.

Importations

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

Code

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

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

Voir l'[exemple complet](#) sur GitHub.

Plus d'informations

- [Instructions relatives à l'utilisation des tableaux](#) dans le guide du Amazon DynamoDB développeur
- [Utilisation des tables DynamoDB dans](#) le guide du Amazon DynamoDB développeur

Utilisation des éléments dans DynamoDB

Dans DynamoDB, un élément est un ensemble d'attributs, chacun étant composé d'un nom et d'une valeur. Une valeur d'attribut peut être de type scalar, set ou document. Pour de plus amples informations, veuillez consulter [Règles de dénomination et types de données](#) dans le Amazon DynamoDB Manuel du développeur

Extraction (Get) d'un élément d'une table

Appelez le module `AmazonDynamoDB.getItem` et passez-lui un [GetItemRequest](#) avec le nom de table et la valeur de la clé primaire de l'élément souhaité. Cette méthode renvoie un objet [GetItemResult](#).

Vous pouvez utiliser la méthode `getItem()` de l'objet `GetItemResult` renvoyé pour récupérer un [mappage](#) des paires de clé (String) et de valeur ([AttributeValue](#)) associées à l'élément.

Importations

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
```

Code

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

key_to_get.put("DATABASE_NAME", new AttributeValue(name));

GetItemRequest request = null;
if (projection_expression != null) {
    request = new GetItemRequest()
        .withKey(key_to_get)
        .withTableName(table_name)
        .withProjectionExpression(projection_expression);
} else {
    request = new GetItemRequest()
        .withKey(key_to_get)
        .withTableName(table_name);
}

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

try {
    Map<String, AttributeValue> returned_item =
        ddb.getItem(request).getItem();
```

```
if (returned_item != null) {
    Set<String> keys = returned_item.keySet();
    for (String key : keys) {
        System.out.format("%s: %s\n",
            key, returned_item.get(key).toString());
    }
} else {
    System.out.format("No item found with the key %s!\n", name);
}
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Ajout d'un nouvel élément à une table

Créez un [mappage](#) des paires clé-valeur qui représentent les attributs de l'élément. Elles doivent inclure les valeurs des champs de clé primaire de la table. Si l'élément identifié par la clé primaire existe déjà, ses champs sont mis à jour par la demande.

Note

Si la table nommée n'existe pas pour vos compte et région, une exception [ResourceNotFoundException](#) est générée.

Importations

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

Code

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

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

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

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Mise à jour d'un élément existant dans une table

Vous pouvez mettre à jour un attribut d'un élément déjà existant dans une table à l'aide de `updateItem`, fournissant un nom de table, une valeur de clé primaire et un mappage des champs à mettre à jour.

Note

Si la table nommée n'existe pas pour vos compte et région, ou si l'élément identifié par la clé primaire que vous avez transmise n'existe pas, une exception [ResourceNotFoundException](#) est levée.

Importations

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



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

Code

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

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

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

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

final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Utilisation de la classe DynamoDBMapper

Le [AWS SDK for Java](#) fournit un [Mapper DynamoDBMapper](#), ce qui vous permet de mapper vos classes côté client aux Amazon DynamoDB tables. Pour utiliser le plugin [Mapper DynamoDBMapper](#), vous définissez la relation entre éléments d'un DynamoDB table et leurs instances d'objet correspondantes dans votre code à l'aide d'annotations (comme indiqué dans l'exemple de code suivant). La classe [DynamoDBMapper](#) vous permet d'accéder à vos tables, d'effectuer diverses opérations de création, de lecture, de mise à jour et de suppression (CRUD), et d'exécuter des requêtes.

Note

La classe [DynamoDBMapper](#) ne vous permet pas de créer, de mettre à jour ni de supprimer des tables.

Importations

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
import com.amazonaws.services.dynamodbv2.model.AmazonDynamoDBException;
```

Code

L'exemple de code Java suivant vous montre comment ajouter du contenu à la table Music en utilisant la classe [DynamoDBMapper](#). Une fois le contenu ajouté à la table, notez qu'un élément est chargé à l'aide des clés de partition et de tri. Ensuite, l'élément Awards est mis à jour. Pour plus d'informations sur la création du Musiquetableau, voir [Création d'une table](#) dans le Amazon DynamoDB Manuel du développeur

```
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.printStackTrace();
    }
}

@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;
}
}
```

Veuillez consulter l'[exemple complet](#) sur GitHub.

Plus d'informations

- [Instructions pour l'utilisation des éléments](#) dans le Amazon DynamoDB Manuel du développeur
- [Utilisation des éléments dans DynamoDB](#) dans le Amazon DynamoDB Manuel du développeur

Amazon EC2 Exemples avec utilisation de l'AWS SDK for Java

Cette section fournit des exemples de programmation [Amazon EC2](#) avec le AWS SDK for Java.

Rubriques

- [Didacticiel : Démarrage d'une instance EC2](#)
- [Utilisation de rôles IAM pour accorder l'accès à AWS Ressources sur Amazon EC2](#)
- [Didacticiel : Amazon EC2 Spot instances](#)
- [Didacticiel : Advanced \(Avancé\) Amazon EC2 Gestion des demandes Spot](#)
- [Gestion des instances Amazon EC2](#)

- [Utilisation d'adresses IP élastiques dans Amazon EC2](#)
- [Utiliser les régions et les zones de disponibilité](#)
- [Utilisation de paires de clés Amazon EC2](#)
- [Utilisation des groupes de sécurité dans Amazon EC2](#)

Didacticiel : Démarrage d'une instance EC2

Ce didacticiel montre comment utiliser l'AWS SDK for Java pour démarrer une instance EC2.

Rubriques

- [Prérequis](#)
- [Création d'un groupe Amazon EC2 de sécurité](#)
- [Créer une paire de clés](#)
- [Exécutez unAmazon EC2Instance](#)

Prérequis

Avant de commencer, assurez-vous d'avoir créé un compte AWS et d'avoir configuré vos informations d'identification AWS. Pour plus d'informations, consultez [Mise en route avec](#).

Création d'un groupe Amazon EC2 de sécurité

EC2-Classic prend sa retraite

Warning

Nous retirons EC2-Classic le 15 août 2022. Nous vous recommandons de migrer d'EC2-Classic vers un VPC. [Pour plus d'informations, consultez la section Migrer d'EC2-Classic vers un VPC dans le guide de l'utilisateur Amazon EC2 ou le guide de l'utilisateur Amazon EC2.](#) Consultez également le billet de blog [EC2-Classic-Classic Networking is Retiring — Here's How to Prepare](#).

Créez un groupe de sécurité faisant office de pare-feu virtuel qui contrôle le trafic réseau d'une ou plusieurs instances EC2. Amazon EC2 associe par défaut vos instances à un groupe de sécurité qui n'autorise aucun trafic entrant. Vous pouvez créer un groupe de sécurité qui autorise vos instances

EC2 à accepter un certain trafic. Par exemple, si vous devez vous connecter à une instance Linux, vous devez configurer le groupe de sécurité afin d'autoriser le trafic SSH. Vous pouvez créer un groupe de sécurité à l'aide de la Amazon EC2 console ou du AWS SDK for Java.

Vous créez un groupe de sécurité à utiliser dans EC2-Classic ou EC2-VPC. Pour plus d'informations sur EC2-Classic et EC2-VPC, consultez la section [Plateformes prises en charge](#) dans le Guide de l' Amazon EC2 utilisateur pour les instances Linux.

Pour plus d'informations sur la création d'un groupe de sécurité à l'aide de la Amazon EC2 console, consultez [Amazon EC2 la section Groupes de sécurité](#) dans le Guide de Amazon EC2 l'utilisateur pour les instances Linux.

1. Créez et initialisez une [CreateSecurityGroupRequest](#) instance. Utilisez la [withGroupName](#) méthode pour définir le nom du groupe de sécurité et la méthode [withDescription](#) pour définir la description du groupe de sécurité, comme suit :

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

Le nom du groupe de sécurité doit être unique dans la AWS région dans laquelle vous initialisez votre Amazon EC2 client. Vous devez utiliser les caractères US-ASCII pour le nom et la description du groupe de sécurité.

2. Transmettez l'objet de demande en tant que paramètre à la [createSecurityGroup](#) méthode. La méthode renvoie un [CreateSecurityGroupResult](#) objet, comme suit :

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

Si vous essayez de créer un groupe de sécurité portant le même nom qu'un groupe de sécurité existant, `createSecurityGroup` lève une exception.

Par défaut, un nouveau groupe de sécurité n'autorise aucun trafic entrant vers votre Amazon EC2 instance. Pour autoriser le trafic entrant, vous devez permettre explicitement l'entrée de groupe de sécurité. Vous pouvez autoriser l'entrée pour des adresses IP individuelles, pour une plage d'adresses IP, pour un protocole spécifique et pour les ports TCP/UDP.

1. Créez et initialisez une [IpPermission](#) instance. Utilisez la méthode [WithIPv4Ranges](#) pour définir la plage d'adresses IP pour laquelle l'entrée doit être autorisée, et utilisez la [withIpProtocol](#) méthode

pour définir le protocole IP. Utilisez les [withToPort](#) méthodes [withFromPort](#) et [withToPort](#) pour spécifier la plage de ports pour lesquels vous souhaitez autoriser l'entrée, comme suit :

```
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);
```

Toutes les conditions spécifiées dans l'objet `IpPermission` doivent être satisfaites pour que l'entrée soit autorisée.

Spécifiez l'adresse IP à l'aide de la notation CIDR. Si vous spécifiez le protocole comme TCP/UDP, vous devez fournir un port source et un port de destination. Vous ne pouvez autoriser les ports que si vous spécifiez TCP ou UDP.

2. Créez et initialisez une [AuthorizeSecurityGroupIngressRequest](#) instance. Utilisez la `withGroupName` méthode pour spécifier le nom du groupe de sécurité et transmettez l'`IpPermission` objet que vous avez initialisé précédemment à la [withIpPermissions](#) méthode, comme suit :

```
AuthorizeSecurityGroupIngressRequest authorizeSecurityGroupIngressRequest =
    new AuthorizeSecurityGroupIngressRequest();

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

3. Passez l'objet de requête dans la méthode [authorizeSecurityGroupIngress](#), comme suit :

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

Si vous appelez `authorizeSecurityGroupIngress` avec des adresses IP pour lesquelles l'entrée est déjà autorisée, la méthode lève une exception. Créez et initialisez un nouvel objet `IpPermission` pour autoriser l'entrée de différentes adresses IP, différents ports et différents protocoles avant d'appeler `AuthorizeSecurityGroupIngress`.

Chaque fois que vous appelez les méthodes d'[authorizeSecurityGroupentrée](#) ou de [authorizeSecurityGroupsortie](#), une règle est ajoutée à votre groupe de sécurité.

Créer une paire de clés

Vous devez indiquer une paire de clés lorsque vous lancez une instance EC2, puis spécifier la clé privée de la paire de clés lorsque vous vous connectez à l'instance. Vous pouvez créer une paire de clés ou utiliser une paire de clés existante que vous avez utilisée lors du lancement d'autres instances. Pour de plus amples informations, veuillez consulter [Amazon EC2 Paires de clés](#) dans le [Amazon EC2 Guide de l'utilisateur](#) pour les instances Linux.

1. Créez et initialisez une instance [CreateKeyPairRequest](#). Utilisez la méthode [withKeyName](#) pour définir le nom de la paire de clés, comme suit :

```
CreateKeyPairRequest createKeyPairRequest = new CreateKeyPairRequest();  
createKeyPairRequest.withKeyName(keyName);
```

Important

Les noms de paire de clés doivent être uniques. Si vous essayez de créer une paire de clés portant le même nom qu'une paire de clés existante, vous obtenez une exception.

2. Transmettez l'objet de requête à la méthode [createKeyPair](#). La méthode envoie une instance [CreateKeyPairResult](#), comme suit :

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

3. Appelez l'objet de résultat de la méthode [getKeyPair](#) pour obtenir un objet [KeyPair](#). Appelez `getKeyPair` l'objet [getKeyMaterial](#) pour obtenir la clé privée codée PEM non chiffrée, comme suit :

```
KeyPair keyPair = new KeyPair();  
keyPair = createKeyPairResult.getKeyPair();  
String privateKey = keyPair.getKeyMaterial();
```


Exécutez unAmazon EC2Instance

Utilisez la procédure suivante pour lancer une ou plusieurs instances EC2 configurés de manière identique à partir de la même AMI (Amazon Machine Image). Une fois que vous avez créé vos instances EC2, vous pouvez vérifier leur statut. Une fois que vos instances EC2 sont en cours d'exécution, vous pouvez vous y connecter.

1. Créez et initialisez une instance [RunInstancesRequest](#). Assurez-vous que l'AMI, la paire de clés et le groupe de sécurité que vous spécifiez existent dans la région que vous avez spécifiée lors de la création de l'objet client.

```
RunInstancesRequest runInstancesRequest =
    new RunInstancesRequest();

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

[withImageId](#)

- ID de l'AMI. Pour découvrir comment rechercher des AMI publiques fournies par Amazon ou créer les vôtres, consultez la page [AMI \(Amazon Machine Image\)](#).

[withInstanceType](#)

- Type d'instance compatible avec l'AMI spécifiée. Pour de plus amples informations, veuillez consulter [Types d'instance](#) dans le Amazon EC2 Guide de l'utilisateur pour les instances Linux.

[withMinCount](#)

- Nombre minimal d'instances EC2 à lancer. Si ce nombre est supérieur au nombre d'instances qu'Amazon EC2 peut lancer dans la zone de disponibilité cible, Amazon EC2 ne lance aucune instance.

[withMaxCount](#)

- Nombre maximal d'instances EC2 à lancer. Si ce nombre est supérieur au nombre d'instances qu'Amazon EC2 peut lancer dans la zone de disponibilité cible, Amazon EC2 lance le plus grand nombre d'instances possible au-dessus de MinCount. Vous pouvez lancer entre 1 instance et le nombre maximal d'instances auquel vous êtes autorisé pour le

type d'instance. Pour plus d'informations, consultez [Combien d'instances puis-je exécuter dans Amazon EC2](#) dans le [Amazon EC2 FAQ générale](#).

[withKeyName](#)

- Nom de la paire de clés EC2. Si vous lancez une instance sans spécifier de paire de clés, vous ne pouvez pas vous y connecter. Pour plus d'informations, consultez [Créer une paire de clés](#).

[withSecurityGroups](#)

- Un ou plusieurs groupes de sécurité. Pour de plus amples informations, veuillez consulter [Création d'un Amazon EC2 Security Group](#).

2. Lancez les instances en transmettant l'objet de demande à la méthode [runInstances](#). Cette méthode renvoie un objet [RunInstancesResult](#), comme suit :

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

Une fois que votre instance est en cours d'exécution, vous pouvez vous y connecter à l'aide de votre paire de clés. Pour de plus amples informations, veuillez consulter [Connectez-vous à votre instance Linux](#) dans le [Amazon EC2 Guide de l'utilisateur pour les instances Linux](#).

Utilisation de rôles IAM pour accorder l'accès à AWS Ressources sur Amazon EC2

Toutes les demandes à Amazon Web Services (AWS) doit être signé de manière chiffrée avec des informations d'identification émises par AWS. Vous pouvez utiliser Rôles IAM pour accorder facilement un accès sécurisé à AWS ressources de votre Amazon EC2 instances.

Cette rubrique fournit des informations sur l'utilisation des rôles IAM avec des applications SDK Java s'exécutant sur Amazon EC2. Pour plus d'informations sur les instances IAM, consultez [Rôles IAM pour Amazon EC2](#) dans le [Amazon EC2 Guide de l'utilisateur pour les instances Linux](#).

Chaîne de fournisseur par défaut et profils d'instance EC2

Si votre application crée une AWS le client utilisant le constructeur par défaut, le client recherche les informations d'identification à l'aide de la chaîne fournisseur d'informations d'identification par défaut, dans l'ordre suivant :

1. Dans les propriétés système Java : `aws.accessKeyId` et `aws.secretKey`.

2. Dans les variables d'environnement du système : `AWS_ACCESS_KEY_ID` et `AWS_SECRET_ACCESS_KEY`.
3. Dans le fichier d'informations d'identification par défaut (l'emplacement de ce fichier varie en fonction de la plateforme).
4. Informations d'identification délivrées via le service de conteneurs Amazon EC2 si la variable d'environnement `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` est définie et le gestionnaire de sécurité a l'autorisation d'accéder à cette variable.
5. Dans les informations d'identification du profil d'instance situées dans les métadonnées d'instance associées au rôle IAM pour l'instance EC2.
6. Informations d'identification du jeton d'identité web à partir de l'environnement ou du conteneur.

L'étape relative aux informations d'identification du profil d'instance dans la chaîne de fournisseur par défaut est disponible uniquement lorsque vous exécutez votre application sur une instance Amazon EC2, mais elle offre la plus grande facilité d'utilisation et une sécurité optimale lorsque vous utilisez des instances Amazon EC2. Vous pouvez également transmettre une instance [InstanceProfileCredentialsProvider](#) directement au constructeur client pour obtenir des informations d'identification de profil d'instance sans avoir recours à la totalité de la chaîne de fournisseur par défaut.

Par Exemple:

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

Lorsque vous utilisez cette approche, le SDK récupère temporairement les informations d'identification qui disposent des mêmes autorisations que celles associées au rôle IAM qui est associé à l'instance Amazon EC2 dans son profil d'instance. Bien que ces informations d'identification soient temporaires et finiront par expirer, `InstanceProfileCredentialsProvider` les actualise régulièrement pour vous afin que les informations d'identification obtenues continuent d'autoriser l'accès à AWS.

Important

L'actualisation des informations d'identification a lieu uniquement lorsque vous utilisez le constructeur client par défaut, qui crée son propre `InstanceProfileCredentialsProvider` dans le cadre de la chaîne

de fournisseur par défaut, ou lorsque vous transmettez une instance `InstanceProfileCredentialsProvider` directement au constructeur client. Si vous utilisez une autre méthode pour obtenir ou transmettre des informations d'identification de profil d'instance, il vous incombe de les vérifier et d'actualiser des informations d'identification expirées.

Si le constructeur client ne peut pas trouver d'informations d'identification à l'aide de la chaîne de fournisseur d'informations d'identification, il lève une exception [AmazonClientException](#).

Procédure : Utilisation de rôles IAM pour les instances EC2

La procédure suivante vous montre comment extraire un objet depuis Amazon S3 Utilisation d'un rôle IAM pour gérer l'accès.

Créer un rôle IAM

Créez un rôle IAM qui accorde l'accès en lecture seule à Amazon S3.

1. Ouvrez la [console IAM](#).
2. Dans le panneau de navigation, sélectionnez Rôles, puis Créer un rôle.
3. Saisissez un nom pour le rôle, puis sélectionnez Étape suivante. Mémorisez ce nom, car il vous sera utile lorsque vous lancerez votre instance Amazon EC2
4. Dans la page Sélectionner un type de rôle page, sous Service AWS Rôles, sélectionnez Amazon EC2 .
5. Dans la page Réglez les autorisations page, sous Sélectionner un modèle de stratégie, sélectionnez Amazon S3 Accès en lecture seule, puis Étape suivante.
6. Sur la page Vérification, sélectionnez Créer un rôle.

Lancement d'une instance EC2 et spécification de votre rôle IAM

Vous pouvez lancer un Amazon EC2 avec un rôle IAM à l'aide de l'Amazon EC2 ou la console AWS SDK for Java.

- Pour lancer un Amazon EC2 à l'aide de la console, suivez les instructions fournies dans [Démarrer avec Amazon EC2 Instances Linux](#) dans le Amazon EC2 Guide de l'utilisateur pour les instances Linux.

Lorsque vous atteignez la page Examiner le lancement de l'instance, sélectionnez Modifier les détails de l'instance. Dans Rôle IAM, choisissez le rôle IAM que vous avez créé précédemment. Exécutez la procédure comme indiqué.

Note

Vous devrez créer ou utiliser un groupe de sécurité existant et une paire de clés pour vous connecter à l'instance.

- Pour lancer un Amazon EC2 avec un rôle IAM à l'aide de l'AWS SDK for Java, voir [Exécutez un Amazon EC2 Instance](#).

Création de votre application

Nous allons créer l'exemple d'application à exécuter sur l'instance EC2. Tout d'abord, créez un répertoire que vous pouvez utiliser pour stocker les fichiers du didacticiel (par exemple, GetS3ObjectApp).

Ensuite, copiez les bibliothèques AWS SDK for Java dans le répertoire nouvellement créé. Si vous avez téléchargé le kit AWS SDK for Java dans votre répertoire ~/Downloads, vous pouvez le copier à l'aide des commandes suivantes :

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

Ouvrez un nouveau fichier, appelez-le GetS3Object.java et ajoutez le code suivant :

```
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();
}
}
```

Ouvrez un nouveau fichier, appelez-le `build.xml` et ajoutez les lignes suivantes :

```
<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>
```

Créez et exécutez le programme modifié. Notez qu'aucune information d'identification n'est stockée dans le programme. Par conséquent, à moins que vous n'ayez votre AWS Les informations d'identification déjà spécifiées, le code lancera `AmazonServiceException`. Par Exemple:

```
$ 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
```

Transfert du programme compilé à votre instance EC2

Transférez le programme à votre instance Amazon EC2 à l'aide d'une copie sécurisée (`scp`), ainsi que les bibliothèques AWS SDK for Java. La séquence de commandes ressemble à ce qui suit.

```
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

En fonction de la distribution Linux que vous avez utilisée, le nom d'utilisateur peut être « `ec2-user` », « `root` » ou « `ubuntu` ». Pour obtenir le nom DNS public de votre instance, ouvrez la

[console EC2](#) et recherchez la valeur de DNS public dans l'onglet Description (par exemple, `ec2-198-51-100-1.compute-1.amazonaws.com`).

Dans les commandes précédentes :

- `GetS3Object.class` est votre programme compilé
- `build.xml` est le fichier ant utilisé pour créer et exécuter votre programme
- les répertoires `lib` et `third-party` sont les dossiers de bibliothèque correspondants du kit AWS SDK for Java.
- Le commutateur `-r` indique que `scp` doit effectuer une copie récursive de tout le contenu des répertoires `library` et `third-party` de la distribution AWS SDK for Java.
- Le commutateur `-p` indique que `scp` doit conserver les autorisations des fichiers sources lorsque ceux-ci sont copiés vers la destination.

Note

Le `-p`Switch fonctionne uniquement sous Linux, macOS ou Unix. Si vous copiez des fichiers à partir de Windows, vous devrez peut-être corriger les autorisations de fichiers sur votre instance à l'aide de la commande suivante :

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

Exécution de l'exemple de programme sur l'instance EC2

Pour exécuter le programme, connectez-vous à votre instance Amazon EC2. Pour de plus amples informations, veuillez consulter [Connectez-vous à votre instance Linux](#) dans le Amazon EC2 Guide de l'utilisateur pour les instances Linux.

Si `ant` n'est pas disponible sur votre instance, installez-le à l'aide de la commande suivante :

```
sudo yum install ant
```

Exécutez ensuite le programme en utilisant `ant` comme suit :

```
ant run
```


Le programme écrit le contenu de votre objet Amazon S3 dans la fenêtre de commande.

Didacticiel : Amazon EC2 Spot instances

Présentation

Les instances Spot vous permettent de placer une offre sur des capacités Amazon Elastic Compute Cloud (Amazon EC2) inutilisées d'une valeur égale à 90 % au maximum du prix d'une instance à la demande et d'exécuter les instances acquises aussi longtemps que votre offre dépasse le prix Spot actuel. Amazon EC2 modifie régulièrement le prix Spot en fonction de l'offre et de la demande, et les clients dont les offres sont égales ou supérieures à ce prix ont accès aux instances Spot disponibles. Tout comme les instances à la demande et les instances réservées, les instances Spot vous offrent une autre possibilité d'obtenir des capacités de calcul supplémentaires.

Les instances Spot peuvent faire baisser de manière significative vos coûts Amazon EC2 pour les traitements par lot, la recherche scientifique, le traitement des images, l'encodage vidéo, l'indexation des données et des sites web, les analyses financières et les tests. De plus, les instances Spot vous donnent accès à de grandes quantités de capacité supplémentaire lorsque le besoin de capacité n'est pas urgent.

Pour utiliser des instances Spot, créez une demande d'instance Spot indiquant le prix maximum que vous êtes prêt à payer par heure d'instance. Cette valeur constitue votre offre. Si votre offre est supérieure au prix Spot actuel, votre demande est satisfaite et vos instances s'exécutent jusqu'à ce que vous décidiez de les résilier ou jusqu'à ce que le prix Spot devienne supérieur à votre offre, selon la première échéance.

Veillez noter les points importants suivants :

- Vous payez souvent un tarif par heure inférieur à celui de votre offre. Amazon EC2 ajuste régulièrement le prix Spot au fur et à mesure que les demandes arrivent et que les offres disponibles changent. Chacun paie le même prix Spot pour cette période, même si l'offre soumise était supérieure. Par conséquent, vous pouvez payer moins que votre offre, mais vous ne paierez jamais plus que votre offre.
- Si vous exécutez des instances Spot et que votre offre n'est plus égale au prix Spot actuel ou ne le dépasse plus, vos instances sont résiliées. Cela signifie que vous devez vous assurer que vos charges de travail et applications sont suffisamment flexibles pour tirer parti de cette capacité opportuniste.

Les instances Spot fonctionnent exactement comme les autres instances Amazon EC2 et, tout comme les autres instances Amazon EC2, elles peuvent être résiliées lorsque vous n'en avez plus besoin. Si vous résiliez votre instance, vous êtes facturé pour toute heure d'utilisation partielle (comme c'est le cas pour les instances à la demande et les instances réservées). Toutefois, si le prix Spot devient supérieur à votre offre et qu'Amazon EC2 résilie votre instance, vous n'êtes pas facturé pour l'heure d'utilisation partielle.

Ce didacticiel explique comment utiliser le kit AWS SDK for Java pour effectuer les tâches suivantes :

- Soumettre une demande Spot
- Déterminer à quel moment la demande Spot est satisfaite
- Annuler la demande Spot
- Résilier les instances associées

Prérequis

Pour utiliser ce didacticiel, vous devez avoir le kit AWS SDK for Java installé, ainsi que satisfaire à ses conditions d'installation prérequis de base. Voir [Configuration de l'AWS SDK for Java](#) pour plus d'informations, consultez.

Étape 1 : Définition de vos informations d'identification

Pour utiliser cet exemple de code, vous devez configurer AWS Informations d'identification . Voir [Configuration AWS Informations d'identification et région pour le développement](#) pour obtenir des instructions sur la façon d'effectuer cette opération.

Note

Nous vous recommandons d'utiliser les informations d'identification d'un utilisateur IAM pour fournir ces valeurs. Pour de plus amples informations, veuillez consulter [S'inscrire à AWS](#) et [Créer un utilisateur IAM](#).

Maintenant que vous avez configuré vos paramètres, vous pouvez commencer à utiliser le code de l'exemple.

Étape 2 : Configuration d'un groupe de sécurité

Un groupe de sécurité fonctionne comme un pare-feu qui contrôle le trafic autorisé en entrée et en sortie d'un groupe d'instances. Par défaut, une instance est lancée sans aucun groupe de sécurité, ce qui signifie que l'ensemble du trafic IP entrant, sur n'importe quel port TCP, est refusé. Par conséquent, avant de soumettre notre demande Spot, nous allons configurer un groupe de sécurité qui permet le trafic réseau nécessaire. Dans le cadre de ce didacticiel, nous allons créer un groupe de sécurité appelé « GettingStarted », qui autorise le trafic SSH (Secure Shell) depuis l'adresse IP à partir de laquelle vous exécutez votre application. Pour configurer un nouveau groupe de sécurité, vous devez inclure ou exécuter l'exemple de code suivant qui configure le groupe de sécurité par programmation.

Après avoir créé un objet client AmazonEC2, nous créons un objet `CreateSecurityGroupRequest` avec le nom « GettingStarted » et une description du groupe de sécurité. Ensuite, nous appelons l'API `ec2.createSecurityGroup` pour créer le groupe.

Pour permettre l'accès au groupe, nous créons un objet `ipPermission` avec la plage d'adresses IP définie sur la représentation CIDR du sous-réseau de l'ordinateur local ; le suffixe « /10 » de l'adresse IP indique le sous-réseau de l'adresse IP spécifiée. Nous configurons également l'objet `ipPermission` avec le protocole TCP et le port 22 (SSH). La dernière étape consiste à appeler `ec2.authorizeSecurityGroupIngress` avec le nom de notre groupe de sécurité et l'objet `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());
}
```

Notez que vous avez uniquement besoin d'exécuter cette application une seule fois pour créer un nouveau groupe de sécurité.

Vous pouvez aussi créer le groupe de sécurité avec AWS Toolkit for Eclipse. Voir [Gestion des groupes de sécurité à partir AWS Cost Explorer](#) pour plus d'informations, consultez.

Étape 3 : Soumission de votre demande Spot

Pour soumettre une demande Spot, vous devez d'abord déterminer le type d'instance, l'AMI (Amazon Machine Image) et le prix maximum de l'offre à utiliser. Vous devez également inclure le groupe de sécurité configuré précédemment, afin de pouvoir vous connecter à l'instance, le cas échéant.

Il existe plusieurs types d'instances au choix. Amazon EC2 Types d'instances pour obtenir la liste complète. Dans le cadre de ce didacticiel, nous allons utiliser t1.micro, le type d'instance le moins cher disponible. Ensuite, nous allons déterminer le type d'AMI à utiliser. Nous allons utiliser ami-a9d09ed1, l'AMI Amazon Linux la plus récente disponible au moment de la rédaction de ce didacticiel. L'AMI la plus récente peut changer au fil du temps, mais vous pouvez déterminer la dernière version de l'AMI en procédant comme suit :

1. Ouvrez la [console Amazon EC2](#).
2. Choisissez le bouton Lancer une instance.
3. La première fenêtre affiche les AMI disponibles. L'ID de l'AMI est répertorié en regard de chaque titre d'AMI. Vous pouvez aussi utiliser l'API DescribeImages, mais l'exploitation de cette commande n'entre pas dans le cadre de ce didacticiel.

Il existe de nombreuses façons d'aborder les offres relatives aux instances Spot. Pour obtenir une vue d'ensemble des diverses approches possibles, regardez la vidéo présentant les [offres relatives aux instances Spot](#). Toutefois, pour commencer, nous allons décrire trois stratégies courantes : offre garantissant un coût inférieur à la tarification à la demande, offre basée sur la valeur du calcul résultant et offre visant à acquérir une capacité de calcul aussi vite que possible.

- Réduction du coût sous la tarification à la demande Vous avez une tâche de traitement par lot dont l'exécution prendra un certain nombre d'heures ou de jours. Toutefois, vous êtes flexible quant aux dates et heures de début et de fin de la tâche. Vous voulez savoir si vous pouvez exécuter cette tâche à un coût inférieur à celui obtenu avec les instances à la demande. Vous examinez l'historique des prix Spot pour les types d'instances à l'aide de la AWS Management Console ou de l'API Amazon EC2. Pour plus d'informations, consultez [Historique de tarification des instance Spots](#). Une fois que vous avez analysé l'historique des prix pour le type d'instance souhaité dans une zone de disponibilité donnée, deux approches sont possibles :
 - Vous pouvez faire une offre à la limite supérieure de la plage de prix Spot (qui sont toujours inférieurs au prix à la demande), en anticipant que votre demande d'instance Spot sera très probablement satisfaite et que vous aurez suffisamment de temps de calcul consécutifs pour terminer la tâche.
 - Vous pouvez également spécifier le montant que vous êtes disposé à payer pour les instances Spot sous forme de % du prix des instances à la demande et prévoir de combiner de nombreuses instances lancées au fil du temps via une demande persistante. Si le prix spécifié est dépassé, l'instance Spot est résiliée. (Nous vous expliquerons comment automatiser cette tâche plus tard dans ce didacticiel.)

- Paiement égal ou inférieur à la valeur du résultat Vous avez une tâche de traitement de données à exécuter. Vous connaissez la valeur des résultats de la tâche suffisamment bien pour savoir qu'ils représentent un réel intérêt en termes de coûts informatiques. Une fois que vous avez analysé l'historique des prix Spot pour votre type d'instance, vous choisissez un prix d'offre pour lequel le coût du temps de calcul n'est pas supérieur à la valeur des résultats de la tâche. Vous créez une offre persistante et faites en sorte qu'elle s'exécute de façon intermittente selon que le prix Spot est égal ou inférieur à votre offre.
- Acquisition rapide de capacité de calcul Vous avez un besoin de capacité supplémentaire imprévu, à court terme, que les instances à la demande ne peuvent pas satisfaire. Une fois que vous avez analysé l'historique des prix Spot pour votre type d'instance, vous effectuez une offre supérieure au prix historique le plus élevé afin d'être quasiment sûr que votre demande sera satisfaite rapidement et que vous pourrez poursuivre les calculs jusqu'à ce qu'elle soit terminée.

Une fois que vous avez choisi votre prix d'offre, vous êtes prêt à demander une instance Spot. Dans le cadre de ce didacticiel, nous allons faire une offre au prix à la demande (0,03 USD) afin de maximiser les chances que votre offre soit satisfaite. Vous pouvez déterminer les types d'instances disponibles et les prix à la demande des instances en accédant à la [Amazon EC2 Page de tarification](#). Vous payez le prix Spot en vigueur pendant la durée d'exécution de vos instances Spot. Les prix Spot sont définis par Amazon EC2 et ajustés graduellement en fonction des tendances à long terme en matière d'offre et de demande de capacité d'instance Spot. Vous pouvez également spécifier le montant que vous êtes disposé à payer pour une instance Spot sous forme de % du prix d'une instance à la demande. Pour demander une instance Spot, il vous suffit de créer votre demande avec les paramètres que vous avez choisis précédemment. Nous allons commencer par créer un objet `RequestSpotInstanceRequest`. L'objet de la demande nécessite le nombre d'instances que vous voulez démarrer et le prix de l'offre. De plus, vous devez définir l'élément `LaunchSpecification` pour la demande, qui comprend le type d'instance, l'ID d'AMI et le groupe de sécurité que vous souhaitez utiliser. Une fois la demande remplie, vous appelez la méthode `requestSpotInstances` sur l'objet `AmazonEC2Client`. L'exemple suivant indique comment demander une instance Spot.

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

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

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

```
requestRequest.setInstanceCount(Integer.valueOf(1));

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

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

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

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

L'exécution de ce code lance une nouvelle demande d'instance Spot. Il existe d'autres options que vous pouvez utiliser pour configurer vos demandes Spot. Pour en savoir plus, rendez-vous sur [Didacticiel : Advanced \(Avancé\) Amazon EC2 Gestion des demandes Spot](#) ou le [RequestSpotInstances](#) dans la classe AWS SDK for Java Référence API.

Note

Vous serez facturé pour toute instance Spot réellement lancée. Veillez donc à annuler toute demande inutile et à résilier toutes les instances que vous lancez afin de réduire les frais associés.

Étape 4 : Détermination de l'état de votre demande Spot

Nous voulons créer le code en attendant que la demande Spot atteigne le statut « actif » avant de passer à la dernière étape. Pour déterminer l'état de la requête Spot, nous interrogeons la méthode [describeSpotInstanceRequests](#) concernant l'état de l'ID de la requête Spot qui nous intéresse.

L'ID de demande créé à l'étape 2 est intégré dans la réponse à notre demande `requestSpotInstances`. L'exemple de code suivant montre comment collecter des ID de

demande à partir de la réponse `requestSpotInstances` et les utiliser pour remplir une `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());
}
```

Pour surveiller votre ID de demande, appelez la méthode `describeSpotInstanceRequests` pour déterminer l'état de la demande. Ensuite, poursuivez la boucle jusqu'à ce que la demande ne soit plus à l'état « ouvert ». Notez que nous recherchons un état autre qu'« ouvert », par exemple « actif », car la demande peut passer directement à l'état « fermé » s'il y a un problème au niveau des arguments de la demande. L'exemple de code suivant décrit comment accomplir cette tâche.

```
// 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);
```

Une fois ce code exécuté, votre demande d'instance Spot est terminée ou a échoué avec une erreur qui s'affiche à l'écran. Dans les deux cas, nous pouvons passer à l'étape suivante pour nettoyer les demandes actives et résilier toutes les instances en cours d'exécution.

Étape 5 : Nettoyage de vos demandes et instances Spot

Nous devons nettoyer nos demandes et instances. Il est important à la fois d'annuler toutes les demandes en cours et de résilier toutes les instances. Si vous annulez simplement vos demandes, cela ne résiliera pas vos instances, ce qui signifie que vous continuerez à payer pour elles. Lorsque vous résiliez vos instances, les demandes Spot peuvent être annulées, mais dans certains cas, par

exemple si vous utilisez les offres persistantes, la résiliation de vos instances ne sera pas suffisante pour empêcher votre demande d'être à nouveau satisfaite. Par conséquent, annuler les offres actives et résilier en même temps toutes les instances en cours d'exécution constitue une bonne pratique.

Le code suivant montre comment annuler vos demandes.

```
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());
}
```

Pour résilier les instances en attente, vous avez besoin de l'ID d'instance associé à la demande qui les a démarrées. L'exemple de code suivant utilise notre code d'origine pour surveiller les instances et ajoute un élément `ArrayList` dans lequel nous stockons l'ID d'instance associé à la réponse `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);
```

Grâce aux ID d'instance stockés dans l'élément `ArrayList`, vous pouvez résilier toutes les instances en cours d'exécution en utilisant l'extrait de code suivant.

```
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());
}
```

Synthèse

Pour résumer, nous fournissons une approche plus orientée objet qui associe les étapes précédemment présentées : initialiser le client EC2, soumettre la demande Spot, déterminer le moment où les demandes Spot ne sont plus à l'état ouvert et nettoyer les demandes Spot persistantes et les instances associées. Nous créons une classe appelée `Requests` qui effectue toutes ces actions.

Nous créons aussi une classe `GettingStartedApp` qui comporte une méthode principale au niveau de laquelle nous effectuons les appels de fonction de haut niveau. Nous initialisons en particulier l'objet `Requests` décrit précédemment. Nous soumettons la demande d'instance Spot. Nous attendons ensuite que l'état de la demande Spot soit « Actif ». Enfin, nous nettoyons les demandes et les instances.

Consultez ou téléchargez le code source complet de cet exemple à partir de [GitHub](#).

Félicitations ! Vous venez de terminer le didacticiel de mise en route permettant de développer le logiciel d'instances Spot avec le kit AWS SDK for Java.

Étapes suivantes

Procédez avec [Didacticiel : Advanced \(Avancé\)Amazon EC2Gestion des demandes Spot](#).

Didacticiel : Advanced (Avancé)Amazon EC2Gestion des demandes Spot

Les instances Spot Amazon EC2 vous permettent de faire une offre sur la capacité Amazon EC2 inutilisée et d'exécuter ces instances, tant que votre offre est supérieure au prix Spot actuel. Amazon EC2 modifie régulièrement le prix Spot en fonction de l'offre et de la demande. Pour plus

d'informations sur les instances Spot, consultez [Instances Spot](#) dans le Amazon EC2 Guide de l'utilisateur pour les instances Linux.

Prérequis

Pour utiliser ce didacticiel, vous devez avoir le kit AWS SDK for Java installé, ainsi que satisfaire à ses conditions d'installation prérequis de base. Voir [Configuration de l'AWS SDK for Java](#) pour plus d'informations, consultez.

Définition de vos informations d'identification

Pour utiliser cet exemple de code, vous devez configurer AWS Informations d'identification . Voir [Configuration AWS Informations d'identification et de la région du développement](#) pour obtenir des instructions sur la façon d'effectuer cette opération.

Note

Nous vous recommandons d'utiliser les informations d'identification d'un IAM pour fournir ces valeurs. Pour de plus amples informations, veuillez consulter [S'inscrire à AWS et créer un IAM User](#).

Maintenant que vous avez configuré vos paramètres, vous pouvez commencer à utiliser le code de l'exemple.

Configuration d'un groupe de sécurité

Un groupe de sécurité fonctionne comme un pare-feu qui contrôle le trafic autorisé en entrée et en sortie d'un groupe d'instances. Par défaut, une instance est lancée sans aucun groupe de sécurité, ce qui signifie que l'ensemble du trafic IP entrant, sur n'importe quel port TCP, est refusé. Par conséquent, avant de soumettre notre demande Spot, nous allons configurer un groupe de sécurité qui permet le trafic réseau nécessaire. Dans le cadre de ce didacticiel, nous allons créer un groupe de sécurité appelé « GettingStarted », qui autorise le trafic SSH (Secure Shell) depuis l'adresse IP à partir de laquelle vous exécutez votre application. Pour configurer un nouveau groupe de sécurité, vous devez inclure ou exécuter l'exemple de code suivant qui configure le groupe de sécurité par programmation.

Après avoir créé un objet client AmazonEC2, nous créons un objet `CreateSecurityGroupRequest` avec le nom « GettingStarted » et une description du groupe de sécurité. Ensuite, nous appelons l'API `ec2.createSecurityGroup` pour créer le groupe.

Pour permettre l'accès au groupe, nous créons un objet `ipPermission` avec la plage d'adresses IP définie sur la représentation CIDR du sous-réseau de l'ordinateur local ; le suffixe « /10 » de l'adresse IP indique le sous-réseau de l'adresse IP spécifiée. Nous configurons également l'objet `ipPermission` avec le protocole TCP et le port 22 (SSH). La dernière étape consiste à appeler `ec2 .authorizeSecurityGroupIngress` avec le nom de notre groupe de sécurité et l'objet `ipPermission`.

(Le code suivant est identique à celui que nous avons utilisé dans le premier didacticiel.)

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

Vous pouvez consulter la totalité de l'exemple de code dans l'exemple de code `advanced.CreateSecurityGroupApp.java`. Notez que vous avez uniquement besoin d'exécuter cette application une seule fois pour créer un nouveau groupe de sécurité.

Note

Vous pouvez aussi créer le groupe de sécurité avec AWS Toolkit for Eclipse. Voir [Gestion des groupes de sécurité dans AWS Cost Explorer](#) dans le AWS Toolkit for Eclipse. Pour plus d'informations, consultez.

Options détaillées de création de demande d'instance Spot

Comme nous l'avons expliqué dans [Didacticiel : Amazon EC2 Instances Spot](#), vous devez construire votre demande avec un type d'instance, une AMI (Amazon Machine Image) et un prix d'offre maximal.

Commençons par créer un objet `RequestSpotInstanceRequest`. L'objet de la demande nécessite le nombre d'instances que vous voulez et le prix de l'offre. De plus, nous devons définir

l'élément `LaunchSpecification` pour la demande, laquelle inclut le type d'instance, l'ID de l'AMI et le groupe de sécurité que vous souhaitez utiliser. Une fois la demande remplie, nous appelons la méthode `requestSpotInstances` sur l'objet `AmazonEC2Client`. Voici un exemple de demande d'une instance Spot.

(Le code suivant est identique à celui que nous avons utilisé dans le premier didacticiel.)

```
// 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);
```

Demandes persistantes et demandes Spot

Lors de la création d'une demande Spot, vous pouvez spécifier plusieurs paramètres facultatifs. Le premier indique si votre demande est persistante ou uniquement ponctuelle. Par défaut, il s'agit d'une demande unique. Une demande unique ne peut être traitée qu'une seule fois, et après que

les instances demandées sont résiliées, la demande est fermée. Une demande persistante est considérée comme devant être traitée chaque fois qu'il n'y a pas d'instance Spot en cours d'exécution pour la même demande. Pour spécifier le type de demande, vous devez simplement définir le type de la demande Spot. Vous pouvez le faire à l'aide du code suivant.

```
// 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);
```

Limitation de la durée d'une demande

Vous pouvez aussi, le cas échéant, spécifier la durée pendant laquelle votre demande demeure valide. Vous pouvez spécifier une heure de début et une heure de fin pour cette période. Par défaut, une demande Spot est considérée comme devant être exécutée à partir du moment où elle est créée jusqu'à ce qu'elle soit achevée ou annulée par vous. Cependant, vous pouvez limiter la période de validité si nécessaire. Un exemple de la façon de spécifier cette période est illustré dans le code suivant.

```
// 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);
```

Regroupement de vos demandes d'instance Spot Amazon EC2

Vous avez la possibilité de regrouper vos demandes d'instances Spot de différentes façons. Nous allons examiner les avantages de l'utilisation de groupes de lancement, de groupes de zones de disponibilité et de groupes de placement.

Si vous voulez vous assurer que vos instances Spot sont toutes lancées et résiliées ensemble, vous avez la possibilité d'exploiter un groupe de lancement. Un groupe de lancement est une étiquette qui regroupe un ensemble d'offres. Toutes les instances d'un groupe de lancement sont démarrées et mises hors service ensemble. Notez que, si les instances d'un groupe de lancement ont déjà été satisfaites, il n'y a aucune garantie que les nouvelles instances lancées avec le même groupe de lancement le soient également. Un exemple de la façon de définir un groupe de lancement est illustré dans l'exemple de code suivant.

```
// 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);
```

Si vous voulez vous assurer que toutes les instances au sein d'une demande sont lancées dans la même zone de disponibilité, et que vous ne vous préoccupez pas de savoir laquelle, vous pouvez exploiter les groupes de zones de disponibilité. Un groupe de zones de disponibilité est une étiquette qui regroupe un ensemble d'instances dans la même zone de disponibilité. Toutes les instances qui partagent un groupe de zones de disponibilité et qui sont satisfaites en même temps démarrent dans la même zone de disponibilité. Voici un exemple de définition d'un groupe de zones de disponibilité.

```
// 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);
```

Vous pouvez spécifier une zone de disponibilité que vous souhaitez pour vos instances Spot. L'exemple de code suivant vous montre comment définir une zone de disponibilité.

```
// 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);
```

Enfin, vous pouvez spécifier un groupe de placement si vous utilisez des instances Spot de Calcul Haute Performance (HPC), telles que les instances de calcul de cluster ou les instances de cluster GPU. Les groupes de placement vous offrent une latence inférieure et une connexion de bande passante élevée entre les instances. Voici un exemple de définition d'un groupe de placement.

```
// 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);
```

Tous les paramètres affichés dans cette section sont facultatifs. Il est également important de souligner que la plupart de ces paramètres, à l'exception du fait que votre offre est unique ou permanente, peuvent réduire la probabilité du traitement de l'offre. Par conséquent, il est important de n'exploiter ces options que si vous en avez besoin. Tous les exemples de code précédents sont regroupés en un seul exemple de code, disponible dans la classe `com.amazonaws.codesamples.advanced.InlineGettingStartedCodeSampleApp.java`.

Comment rendre une partition racine permanente après une interruption ou une mise hors service

L'un des moyens les plus simples de gérer une interruption de vos instances Spot est de vous assurer que vos données sont contrôlées sur un magasin Amazon Elastic Block (Amazon Amazon EBS) sur une cadence régulière. Grâce à un contrôle régulier, en cas d'interruption, vous perdez uniquement les données créées depuis le dernier point de contrôle (en présumant qu'il n'y ait pas eu d'autres actions non idempotentes exécutées entretemps). Pour faciliter le processus, vous pouvez configurer votre demande Spot afin de vous assurer que votre partition racine ne sera pas supprimée lors de l'interruption ou de la résiliation. Nous avons ajouté un nouveau code dans l'exemple suivant, qui montre comment activer ce scénario.

Dans le code ajouté, nous créons un `BlockDeviceMapping` et définissez son objet associé Amazon Elastic Block Store (Amazon EBS) à un `Amazon EBS` objet que nous avons configuré pour ne pas être supprimé si l'instance Spot est terminée. Ensuite, nous ajoutons cet objet `BlockDeviceMapping` à la liste `ArrayList` que nous incluons dans la spécification de lancement.

```
// 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);
```

En supposant que vous vouliez rattacher ce volume à votre instance au démarrage, vous pouvez également utiliser les paramètres de mappage de périphérique de stockage en mode bloc. Si vous avez attaché une partition autre qu'une partition racine, vous pouvez aussi spécifier `AmazonAmazonEBS` à associer à votre instance Spot après sa reprise. Pour ce faire, vous devez simplement spécifier un ID d'instantané (snapshot) dans votre `EbsBlockDevice` et un autre nom d'appareil dans vos objets `BlockDeviceMapping`. En tirant parti des mappages de périphérique de stockage en mode bloc, il peut être plus facile d'amorcer votre instance.

L'utilisation de la partition racine pour contrôler vos données critiques est une excellente façon de gérer le risque d'une interruption de vos instances. Pour plus d'informations sur la gestion des risques d'interruption, consultez la vidéo [Gestion des interruptions](#).

Balises de vos demandes et instances Spot

Ajout de balises à Amazon EC2 peuvent simplifier l'administration de votre infrastructure cloud. Les balises, sorte de métadonnées, peuvent être utilisées pour créer des noms conviviaux, faciliter les recherches et améliorer la coordination entre plusieurs utilisateurs. Vous pouvez également utiliser des balises pour automatiser les scripts et des parties de vos processus. Pour en savoir plus sur le marquage Amazon EC2 ressources, accédez à [Utilisation de balises](#) dans le Amazon EC2 Guide de l'utilisateur pour les instances Linux.

Balises des demandes d'

Pour ajouter des balises à vos demandes Spot, vous devez les baliser après qu'elles ont été demandées. La valeur renvoyée par `requestSpotInstances()` vous fournit un objet [RequestSpotInstancesResult](#), que vous pouvez utiliser pour obtenir l'ID de la requête Spot pour le balisage :

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

Une fois que vous avez les ID, vous pouvez baliser les demandes en ajoutant leurs identifiants à un [CreateTagsRequest](#) et appelle l'intention Amazon EC2 client `createTags()` Méthode :

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

Balilage d'instances

Comme pour les demandes Spot elles-mêmes, vous ne pouvez baliser une instance qu'une fois qu'elle a été créée, ce qui se produit quand la demande Spot a été satisfaite (elle n'est plus à l'état ouvert).

Vous pouvez vérifier le statut de vos demandes en appelant leAmazon EC2client[describeSpotInstanceRequests\(\)](#) avec un[DescribeSpotInstanceRequestsRequest](#) ponctuelleobjet. L'objet [DescribeSpotInstanceRequestsResult](#) renvoyé contient une liste d'objets [SpotInstanceRequest](#) que vous pouvez utiliser pour interroger l'état de vos requêtes Spot et obtenir leurs ID d'instance une fois qu'elles ne sont plus à l'état ouvert.

Une fois que la demande Spot n'est plus ouverte, vous pouvez récupérer son ID d'instance à partir de l'objet [SpotInstanceRequest](#) en appelant sa méthode [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);
```

Maintenant, vous pouvez baliser les instances qui sont renvoyées :

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

Annulation des demandes Spot et mise hors service des instances

Annulation d'une demande Spot

Pour annuler une demande d'instance Spot, appelez `cancelSpotInstanceRequests` sur le `Amazon EC2` client avec un [CancelSpotInstanceRequestsRequest](#) ponctuelle objet.

```
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());
}
```

Mise hors service d'instances Spot

Vous pouvez résilier toutes les instances Spot qui sont en cours d'exécution en transmettant leurs identifiants à l'intention `Amazon EC2` client `terminateInstances()` méthode.

```
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());
}
```

Synthèse

Pour résumer, nous fournissons une approche plus orientée objet qui associe les étapes illustrées dans ce didacticiel en une classe facile à utiliser. Nous instancions une classe appelée `Requests` qui exécute ces actions. Nous créons aussi une classe `GettingStartedApp` qui comporte une méthode principale au niveau de laquelle nous effectuons les appels de fonction de haut niveau.

Consultez ou téléchargez le code source complet de cet exemple à partir de [GitHub](#).

Félicitations ! Vous venez de terminer le didacticiel sur les fonctionnalités de demande avancées permettant de développer un logiciel d'instances Spot avec le kit AWS SDK for Java.

Gestion des instances Amazon EC2

Création d'une instance

Création d'un nouveau Amazon EC2 en appelant l'instance `AmazonEC2Client.runInstances`, en lui fournissant un [RunInstancesRequest](#) contenant le [Amazon Machine Image \(AMI\)](#) à utiliser et [type d'instance](#).

Importations

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

Code

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

RunInstancesResult run_response = ec2.runInstances(run_request);

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

Consultez l'[exemple complet](#).

Démarrage d'une instance

Pour démarrer un objet Amazon EC2, appelez l'instance `AmazonEC2Client.startInstances`, en lui fournissant un [StartInstancesRequest](#) contenant l'ID de l'instance à démarrer.

Importations

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

Code

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

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

ec2.startInstances(request);
```

Consultez l'[exemple complet](#).

Arrêt d'une instance

Pour arrêter un objet Amazon EC2, appelez l'instance Amazon EC2Client stopInstances, en lui fournissant un [StopInstancesRequest](#) contenant l'ID de l'instance à arrêter.

Importations

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

Code

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

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

ec2.stopInstances(request);
```

Consultez l'[exemple complet](#).

Redémarrage d'une instance

Pour redémarrer un objet Amazon EC2, appelez l'instance Amazon EC2Client rebootInstances, en lui fournissant un [RebootInstancesRequest](#) contenant l'ID de l'instance à redémarrer.

Importations

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

Code

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

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

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

Consultez l'[exemple complet](#).

Description des instances

Pour répertorier vos instances, créez un objet [DescribeInstancesRequest](#) et appelez le client Amazon EC2 `describeInstances` méthode. Cette méthode renvoie un objet [DescribeInstancesResult](#) que vous pouvez utiliser pour répertorier les instances Amazon EC2 de vos compte et région.

Les instances sont regroupées par réservation. Chaque réservation correspond à l'appel de `startInstances` qui a lancé l'instance. Pour afficher vos instances, vous devez d'abord appeler la méthode `getReservations` de la classe `DescribeInstancesResult`, puis appeler la méthode `getReservations` method, and then call `getInstances` sur chaque objet [Reservation](#) renvoyé.

Importations

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.Reservation;
```

Code


```
final AmazonEC2 ec2 = AmazonEC2ClientBuilder.defaultClient();
boolean done = false;

DescribeInstancesRequest request = new DescribeInstancesRequest();
while(!done) {
    DescribeInstancesResult response = ec2.describeInstances(request);

    for(Reservation reservation : response.getReservations()) {
        for(Instance instance : reservation.getInstances()) {
            System.out.printf(
                "Found instance with id %s, " +
                "AMI %s, " +
                "type %s, " +
                "state %s " +
                "and monitoring state %s",
                instance.getInstanceId(),
                instance.getImageId(),
                instance.getInstanceType(),
                instance.getState().getName(),
                instance.getMonitoring().getState());
        }
    }

    request.setNextToken(response.getNextToken());

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

Les résultats sont paginés ; vous pouvez obtenir plus de résultats en transmettant la valeur renvoyée par la méthode `getNextToken` de l'objet de résultat à la méthode `setNextToken` de l'objet de la demande d'origine, puis en utilisant le même objet de la demande lors de votre prochain appel de `describeInstances`.

Consultez l'[exemple complet](#).

Surveillance d'une instance

Vous pouvez surveiller les différents aspects de vos instances Amazon EC2, tels que l'utilisation de l'UC et du réseau, la mémoire disponible et l'espace disque restant. Pour plus d'informations

sur la surveillance des instances, consultez [Surveillance Amazon EC2](#) dans le [Amazon EC2 Guide de l'utilisateur pour les instances Linux](#).

Pour commencer à surveiller une instance, vous devez créer un [MonitorInstancesRequest](#) avec l'ID de l'instance à surveiller et le transmettre au client `AmazonEC2Client` `monitorInstances` Méthode.

Importations

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

Code

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

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

ec2.monitorInstances(request);
```

Consultez l'[exemple complet](#).

Arrêt de la surveillance des instances

Pour arrêter la surveillance d'une instance, créez un [UnmonitorInstancesRequest](#) avec l'ID de l'instance dont vous souhaitez arrêter la surveillance et transmettez-le au `unmonitorInstances` Méthode.

Importations

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

Code

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

UnmonitorInstancesRequest request = new UnmonitorInstancesRequest()
```

```
.withInstanceIds(instance_id);  
  
ec2.unmonitorInstances(request);
```

Consultez l'[exemple complet](#).

En savoir plus

- [RunInstances](#) dans le Amazon EC2 API Reference
- [DescribeInstances](#) dans le Amazon EC2 API Reference
- [StartInstances](#) dans le Amazon EC2 API Reference
- [StopInstances](#) dans le Amazon EC2 API Reference
- [RebootInstances](#) dans le Amazon EC2 API Reference
- [MonitorInstances](#) dans le Amazon EC2 API Reference
- [UnmonitorInstances](#) dans le Amazon EC2 API Reference

Utilisation d'adresses IP élastiques dans Amazon EC2

EC2-Classic prend sa retraite

Warning

Nous retirons EC2-Classic le 15 août 2022. Nous vous recommandons de migrer d'EC2-Classic vers un VPC. [Pour plus d'informations, consultez la section Migrer d'EC2-Classic vers un VPC dans le guide de l'utilisateur Amazon EC2 ou le guide de l'utilisateur Amazon EC2.](#) Consultez également le billet de blog [EC2-Classic-Classic Networking is Retiring — Here's How to Prepare](#).

Allocation d'une adresse IP Elastic

Pour utiliser une adresse IP Elastic, commencez par en attribuer une à votre compte, puis associez-la à votre instance ou à une interface réseau.

Pour allouer une adresse IP élastique, appelez la `allocateAddress` méthode `AmazonEC2Client` avec un [AllocateAddressRequest](#) objet contenant le type de réseau (EC2 classique ou VPC).

Le fichier renvoyé [AllocateAddressResult](#) contient un ID d'allocation que vous pouvez utiliser pour associer l'adresse à une instance, en transmettant l'ID d'allocation et l'ID d'instance dans a [AssociateAddressRequest](#) à la méthode du client Amazon EC2. `associateAddress`

Importations

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

Code

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

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

AllocateAddressResult allocate_response =
    ec2.allocateAddress(allocate_request);

String allocation_id = allocate_response.getAllocationId();

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

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

Consultez l'[exemple complet](#).

Description des adresses IP Elastic

Pour répertorier les adresses IP élastiques attribuées à votre compte, appelez la méthode `AmazonEC2Client.describeAddresses` Il renvoie un [DescribeAddressesResult](#) que vous pouvez utiliser pour obtenir une liste d'objets [Address](#) qui représentent les adresses IP élastiques de votre compte.

Importations

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

Code

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

DescribeAddressesResult response = ec2.describeAddresses();

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

Consultez l'[exemple complet](#).

Libération d'une adresse IP Elastic

Pour libérer une adresse IP élastique, appelez la `releaseAddress` méthode `AmazonEC2Client` en lui transmettant un identifiant [ReleaseAddressRequest](#) contenant l'ID d'allocation de l'adresse IP élastique que vous souhaitez libérer.

Importations

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.ReleaseAddressRequest;
import com.amazonaws.services.ec2.model.ReleaseAddressResult;
```

Code

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

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

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

Une fois que vous avez publié une adresse IP élastique, elle est publiée dans le pool d'adresses AWS IP et il se peut que vous ne soyez plus disponible par la suite. Veillez à mettre à jour vos enregistrements DNS, ainsi que les serveurs ou les appareils qui communiquent avec l'adresse. Si vous tentez de libérer une adresse IP élastique que vous avez déjà publiée, un `AuthFailuremessage` d'erreur s'affichera si l'adresse est déjà attribuée à une autre adresse Compte AWS.

Si vous utilisez EC2-Classic ou un VPC par défaut, la libération d'une adresse IP Elastic la dissocie automatiquement de toute instance à laquelle elle est associée. Pour dissocier une adresse IP élastique sans la divulguer, utilisez la méthode `AmazonEC2Client.disassociateAddress`

Si vous utilisez un VPC autre que par défaut, vous devez utiliser `disassociateAddress` pour dissocier l'adresse IP Elastic avant d'essayer de la libérer. Dans le cas contraire, Amazon EC2 renvoie une erreur (`InvalidipAddress. InUse`).

Consultez l'[exemple complet](#).

En savoir plus

- [Adresses IP élastiques](#) dans le guide de Amazon EC2 l'utilisateur pour les instances Linux
- [AllocateAddress](#) dans la référence de Amazon EC2 l'API
- [DescribeAddresses](#) dans la référence de Amazon EC2 l'API
- [ReleaseAddress](#) dans la référence de Amazon EC2 l'API

Utiliser les régions et les zones de disponibilité

Décrire les régions

Pour répertorier les régions disponibles pour votre compte, appelez `describeRegions` Méthode. Cette méthode renvoie un objet [DescribeRegionsResult](#). Appelez la méthode `getRegions` de l'objet renvoyé pour obtenir une liste d'objets [Region](#) qui représentent chaque région.

Importations

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

Code

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

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

Consultez l'[exemple complet](#).

Décrire les zones de disponibilité

Pour répertorier chaque zone de disponibilité disponible pour votre compte, appelez `describeAvailabilityZones` méthode. Cette méthode renvoie un objet [DescribeAvailabilityZonesResult](#). Appelez sa méthode `getAvailabilityZones` pour obtenir une liste d'objets [AvailabilityZone](#) qui représentent chaque zone de disponibilité.

Importations

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

Code

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

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

Consultez l'[exemple complet](#).

Décrire les comptes

Pour décrire votre compte, appelez `describeAccountAttributes` Méthode. Cette méthode renvoie un objet [DescribeAccountAttributesResult](#). Invoquez la méthode `getAccountAttributes` pour obtenir une liste des objets [AccountAttribute](#). Parcourez la liste afin de récupérer un objet [AccountAttribute](#).

Vous pouvez obtenir les valeurs attributaires de votre compte en invoquant le `AccountAttribute`'`getAttributeValues` Méthode. Cette méthode renvoie une liste des objets [AccountAttributeValue](#). Vous pouvez parcourir cette deuxième liste pour afficher la valeur des attributs (voir l'exemple de code ci-dessous).

Importations

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.AccountAttributeValue;
import com.amazonaws.services.ec2.model.DescribeAccountAttributesResult;
import com.amazonaws.services.ec2.model.AccountAttribute;
import java.util.List;
import java.util.ListIterator;
```

Code

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

try{
    DescribeAccountAttributesResult accountResults = ec2.describeAccountAttributes();
    List<AccountAttribute> accountList = accountResults.getAccountAttributes();
}
```



```
for (ListIterator iter = accountList.listIterator(); iter.hasNext(); ) {

    AccountAttribute attribute = (AccountAttribute) iter.next();
    System.out.print("\n The name of the attribute is
"+attribute.getAttributeName());
    List<AccountAttributeValue> values = attribute.getAttributeValues();

    //iterate through the attribute values
    for (ListIterator iterVals = values.listIterator(); iterVals.hasNext(); ) {
        AccountAttributeValue myValue = (AccountAttributeValue) iterVals.next();
        System.out.print("\n The value of the attribute is
"+myValue.getAttributeValue());
    }
}
System.out.print("Done");
}
catch (Exception e)
{
    e.printStackTrace();
}
```

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [Régions et Zones de disponibilité](#) dans le Amazon EC2 Guide de l'utilisateur pour les instances Linux
- [DescribeRegions](#) dans le Amazon EC2 API Reference
- [DescribeAvailabilityZones](#) dans le Amazon EC2 API Reference

Utilisation de paires de clés Amazon EC2

Création d'une paire de clés

Pour créer une key pair, appelez le `createKeyPair` avec un [CreateKeyPairRequest](#) qui contient le nom de la clé.

Importations

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

```
import com.amazonaws.services.ec2.model.CreateKeyPairRequest;
import com.amazonaws.services.ec2.model.CreateKeyPairResult;
```

Code

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

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

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

Consultez l'[exemple complet](#).

Description de paire de clés

Pour répertorier vos paires de clés ou pour obtenir des informations sur celles-ci, appelez `describeKeyPairs` méthode. Cette méthode renvoie un objet [DescribeKeyPairsResult](#) que vous pouvez utiliser pour accéder à la liste des paires de clés en appelant sa méthode `getKeyPairs`, qui renvoie une liste d'objets [KeyPairInfo](#).

Importations

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

Code

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

DescribeKeyPairsResult response = ec2.describeKeyPairs();

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

Consultez l'[exemple complet](#).

Suppression d'une paire de clés

Pour supprimer une key pair, appelez `deleteKeyPair`, en lui transmettant un [DeleteKeyPairRequest](#) qui contient le nom de la key pair à supprimer.

Importations

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

Code

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

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

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

Consultez l'[exemple complet](#).

En savoir plus

- [Amazon EC2 Paires de clés](#) dans le Amazon EC2 Guide de l'utilisateur pour les instances Linux
- [CreateKeyPair](#) dans le Amazon EC2 API Reference
- [DescribeKeyPairs](#) dans le Amazon EC2 API Reference
- [DeleteKeyPair](#) dans le Amazon EC2 API Reference

Utilisation des groupes de sécurité dans Amazon EC2

Création d'un groupe de sécurité

Pour créer un groupe de sécurité, appelez la `createSecurityGroup` méthode Amazon EC2 Client avec un nom [CreateSecurityGroupRequest](#) contenant le nom de la clé.

Importations

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.CreateSecurityGroupRequest;
import com.amazonaws.services.ec2.model.CreateSecurityGroupResult;
```

Code

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

CreateSecurityGroupRequest create_request = new
    CreateSecurityGroupRequest()
        .withGroupName(group_name)
        .withDescription(group_desc)
        .withVpcId(vpc_id);

CreateSecurityGroupResult create_response =
    ec2.createSecurityGroup(create_request);
```

Consultez l'[exemple complet](#).

Configuration d'un groupe de sécurité

Un groupe de sécurité peut contrôler le trafic entrant et le trafic sortant de vos instances Amazon EC2.

Pour ajouter des règles d'entrée à votre groupe de sécurité, utilisez la `authorizeSecurityGroupIngress` méthode Amazon EC2Client en indiquant le nom du groupe de sécurité et les règles d'accès ([IpPermission](#)) que vous souhaitez lui attribuer dans un [AuthorizeSecurityGroupIngressRequest](#) objet. L'exemple suivant montre comment ajouter des autorisations IP à un groupe de sécurité.

Importations

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.CreateSecurityGroupRequest;
import com.amazonaws.services.ec2.model.CreateSecurityGroupResult;
```

Code

```
IpRange ip_range = new IpRange()
    .withCidrIp("0.0.0.0/0");

IpPermission ip_perm = new IpPermission()
    .withIpProtocol("tcp")
    .withToPort(80)
    .withFromPort(80)
    .withIpv4Ranges(ip_range);

IpPermission ip_perm2 = new IpPermission()
    .withIpProtocol("tcp")
    .withToPort(22)
    .withFromPort(22)
    .withIpv4Ranges(ip_range);

AuthorizeSecurityGroupIngressRequest auth_request = new
    AuthorizeSecurityGroupIngressRequest()
        .withGroupName(group_name)
        .withIpPermissions(ip_perm, ip_perm2);

AuthorizeSecurityGroupIngressResult auth_response =
    ec2.authorizeSecurityGroupIngress(auth_request);
```

Pour ajouter une règle de sortie au groupe de sécurité, fournissez des données similaires dans une [AuthorizeSecurityGroupEgressRequest](#) `authorizeSecurityGroupEgress` méthode Amazon EC2Client.

Consultez l'[exemple complet](#).

Description des groupes de sécurité

Pour décrire vos groupes de sécurité ou obtenir des informations à leur sujet, appelez la `describeSecurityGroups` méthode Amazon EC2Client. Elle renvoie un [DescribeSecurityGroupsResult](#) que vous pouvez utiliser pour accéder à la liste des groupes de sécurité en appelant la `getSecurityGroups` méthode, qui renvoie une liste d'[SecurityGroup](#) objets.

Importations

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

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

Code

```
final String USAGE =
    "To run this example, supply a group id\n" +
    "Ex: DescribeSecurityGroups <group-id>\n";

if (args.length != 1) {
    System.out.println(USAGE);
    System.exit(1);
}

String group_id = args[0];
```

Consultez l'[exemple complet](#).

Suppression d'un groupe de sécurité

Pour supprimer un groupe de sécurité, appelez la `deleteSecurityGroup` méthode Amazon EC2Client en lui transmettant une [DeleteSecurityGroupRequest](#) adresse contenant l'ID du groupe de sécurité à supprimer.

Importations

```
import com.amazonaws.services.ec2.AmazonEC2;
import com.amazonaws.services.ec2.AmazonEC2ClientBuilder;
import com.amazonaws.services.ec2.model.DeleteSecurityGroupRequest;
import com.amazonaws.services.ec2.model.DeleteSecurityGroupResult;
```

Code

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

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

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

Consultez l'[exemple complet](#).

En savoir plus

- [Amazon EC2 Groupes de sécurité](#) dans le guide de Amazon EC2 l'utilisateur pour les instances Linux
- [Autorisation du trafic entrant pour vos instances Linux](#) dans le guide de l'Amazon EC2 utilisateur pour les instances Linux
- [CreateSecurityGroup](#) dans la référence de Amazon EC2 l'API
- [DescribeSecurityGroups](#) dans la référence de Amazon EC2 l'API
- [DeleteSecurityGroup](#) dans la référence de Amazon EC2 l'API
- [AuthorizeSecurityGroupIngress](#) dans la référence de Amazon EC2 l'API

Exemples IAM utilisant le module AWS SDK for Java

Cette section fournit des exemples de programmation d'[IAM](#) à l'aide du kit [AWS SDK for Java](#).

AWS Identity and Access Management (IAM) vous permet de contrôler de façon sécurisée l'accès à AWS services et ressources pour vos utilisateurs. Avec IAM, vous pouvez créer et gérer AWS utilisateurs et groupes, et utiliser les autorisations pour autoriser ou refuser leur accès aux AWS. Pour un guide complet sur IAM, consultez le site [IAM Guide de l'utilisateur](#).

Note

Les exemples incluent uniquement le code nécessaire pour démontrer chaque technique. L'exemple de code complet [est disponible sur GitHub](#). À partir de là, vous pouvez télécharger un fichier source unique ou cloner le référentiel en local pour obtenir tous les exemples à générer et exécuter.

Rubriques

- [Gestion des clés d'accès IAM](#)
- [Gestion des utilisateurs IAM](#)
- [Utilisation des alias de compte IAM](#)
- [Utilisation des stratégies IAM](#)
- [Utilisation des certificats de serveur IAM](#)

Gestion des clés d'accès IAM

Création d'une clé d'accès

Pour créer une clé d'accès IAM, appelez `AmazonIdentityManagementClient.createAccessKey` avec une méthode [CreateAccessKeyRequest](#) objet.

`CreateAccessKeyRequest` possède deux constructeurs, l'un prend un nom d'utilisateur et l'autre ne prend aucun paramètre. Si vous utilisez la version qui ne prend aucun paramètre, vous devez définir le nom d'utilisateur à l'aide de la méthode setter `withUserName` avant de transmettre celui-ci à la méthode `createAccessKey`.

Importations

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateAccessKeyRequest;
import com.amazonaws.services.identitymanagement.model.CreateAccessKeyResult;
```

Code

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

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Affichage de la liste des clés d'accès

Pour répertorier les clés d'accès d'un utilisateur donné, créez un [ListAccessKeysRequest](#) contenant le nom de l'utilisateur dont vous devez répertorier les clés et le transmettre à `AmazonIdentityManagementClient.listAccessKeys` méthode.

Note

Si vous ne fournissez pas de nom d'utilisateur à `listAccessKeys`, il tentera de répertorier les clés d'accès associées à la Compte AWS qui a signé la demande.

Importations

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.AccessKeyMetadata;
import com.amazonaws.services.identitymanagement.model.ListAccessKeysRequest;
import com.amazonaws.services.identitymanagement.model.ListAccessKeysResult;
```

Code

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

boolean done = false;
ListAccessKeysRequest request = new ListAccessKeysRequest()
    .withUserName(username);

while (!done) {

    ListAccessKeysResult response = iam.listAccessKeys(request);

    for (AccessKeyMetadata metadata :
        response.getAccessKeyMetadata()) {
        System.out.format("Retrieved access key %s",
            metadata.getAccessKeyId());
    }

    request.setMarker(response.getMarker());

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

Les résultats de `listAccessKeys` sont paginés par défaut (avec un maximum de 100 enregistrements par appel). Vous pouvez appeler `getIsTruncated` sur l'objet [ListAccessKeysResult](#) renvoyé pour voir si la requête a renvoyé moins de résultats que ce qui est disponible. Si tel est le cas, appelez `setMarker` sur l'objet `ListAccessKeysRequest` et retransmettez-le dans le prochain appel de `listAccessKeys`.

Veillez consulter l'[exemple complet](#) sur GitHub.

Récupération de l'heure de la dernière utilisation d'une clé d'accès

Pour obtenir l'heure à laquelle une clé d'accès a été utilisée en dernier, appelez `getAccessKeyLastUsed` avec l'ID de la clé d'accès (qui peut être transmis à l'aide d'un [GetAccessKeyLastUsedRequest](#) ou directement à la surcharge qui prend directement l'ID de clé d'accès).

Vous pouvez ensuite utiliser l'objet [GetAccessKeyLastUsedResult](#) renvoyé pour récupérer l'heure de la dernière utilisation de la clé.

Importations

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

Code

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

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

GetAccessKeyLastUsedResult response = iam.getAccessKeyLastUsed(request);

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Activation ou désactivation des clés d'accès

Vous pouvez activer ou désactiver une clé d'accès en créant un [UpdateAccessKeyRequest](#), fournissant l'ID de clé d'accès, éventuellement le nom d'utilisateur et le nom souhaité [État](#), puis transmet l'objet de demande à `AmazonIdentityManagementClient.updateAccessKey` méthode.

Importations

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

Code

```
final AmazonIdentityManagement iam =  
    AmazonIdentityManagementClientBuilder.defaultClient();  
  
UpdateAccessKeyRequest request = new UpdateAccessKeyRequest()  
    .withAccessKeyId(access_id)  
    .withUserName(username)  
    .withStatus(status);  
  
UpdateAccessKeyResult response = iam.updateAccessKey(request);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Suppression d'une clé d'accès

Pour supprimer définitivement une clé d'accès, appelez `deleteKey`, en lui fournissant un [DeleteAccessKeyRequest](#) contenant l'ID et le nom d'utilisateur de la clé d'accès.

Note

Une fois supprimée, une clé ne peut plus être récupérée ou utilisée. Pour désactiver temporairement une clé de façon à pouvoir la réactiver ultérieurement, utilisez plutôt la méthode [updateAccessKey](#).

Importations

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

Code

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

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [CreateAccessKey](#) dans la Référence d'API IAM
- [ListAccessKeys](#) dans la Référence d'API IAM
- [GetAccessKeyLastUsed](#) dans la Référence d'API IAM
- [UpdateAccessKey](#) dans la Référence d'API IAM
- [DeleteAccessKey](#) dans la Référence d'API IAM

Gestion des utilisateurs IAM

Création d'un utilisateur

Créez un nouvel utilisateur IAM en fournissant le nom d'utilisateur à la `AmazonIdentityManagementClient.createUser` méthode, soit directement, soit à l'aide d'un [CreateUserRequest](#) objet contenant le nom d'utilisateur.

Importations

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

```
import com.amazonaws.services.identitymanagement.model.CreateUserRequest;
import com.amazonaws.services.identitymanagement.model.CreateUserResult;
```

Code

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

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

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

Voir l'[exemple complet](#) sur GitHub.

Affichage d'une liste d'utilisateurs

Pour répertorier les utilisateurs IAM de votre compte, créez-en un nouveau [ListUsersRequest](#) et transmettez-le à `AmazonIdentityManagementClient` la `listUsers` méthode. Vous pouvez récupérer la liste des utilisateurs en appelant `getUsers` l'[ListUsersResult](#) objet renvoyé.

La liste d'utilisateurs renvoyée par `listUsers` est paginée. Vous pouvez vérifier s'il existe plus de résultats à récupérer en appelant la méthode `getIsTruncated` de l'objet de réponse. Si celle-ci renvoie `true`, appelez la méthode `setMarker()` de l'objet de demande, en lui transmettant la valeur de retour de la méthode `getMarker()` de l'objet de réponse.

Importations

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

Code

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

boolean done = false;
```

```
ListUsersRequest request = new ListUsersRequest();

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

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

    request.setMarker(response.getMarker());

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

Voir l'[exemple complet](#) sur GitHub.

Mise à jour d'un utilisateur

Pour mettre à jour un utilisateur, appelez la `updateUser` méthode de l'`AmazonIdentityManagementClient` objet, qui utilise un [UpdateUserRequest](#) objet que vous pouvez utiliser pour modifier le nom ou le chemin de l'utilisateur.

Importations

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.UpdateUserRequest;
import com.amazonaws.services.identitymanagement.model.UpdateUserResult;
```

Code

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

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

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

Voir l'[exemple complet](#) surGitHub.

Suppression d'un utilisateur

Pour supprimer un utilisateur, appelez `AmazonIdentityManagementClient` `sdeleteUser` demande à l'aide d'un [UpdateUserRequest](#) objet défini avec le nom d'utilisateur à supprimer.

Importations

```
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.DeleteConflictException;
import com.amazonaws.services.identitymanagement.model.DeleteUserRequest;
```

Code

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

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

try {
    iam.deleteUser(request);
} catch (DeleteConflictException e) {
    System.out.println("Unable to delete user. Verify user is not" +
        " associated with any resources");
    throw e;
}
```

Voir l'[exemple complet](#) surGitHub.

En savoir plus

- [Les utilisateurs d'IAM](#) dans le guide deIAM l'utilisateur
- [Gestion des utilisateurs IAM](#) dans le guide deIAM l'utilisateur
- [CreateUser](#) dans la référence de l'API IAM
- [ListUsers](#) dans la référence de l'API IAM
- [UpdateUser](#) dans la référence de l'API IAM
- [DeleteUser](#) dans la référence de l'API IAM

Utilisation des alias de compte IAM

Si vous voulez que l'URL de votre page de connexion contienne le nom de votre société ou tout autre identifiant convivial au lieu de votre Compte AWS ID d', vous pouvez créer un alias pour ce dernier Compte AWS.

Note

AWSprend en charge exactement un alias de compte par compte.

Création d'un alias de compte

Pour créer un alias de compte, appelez la `createAccountAlias` méthode `AmazonIdentityManagementClient`'s avec un [CreateAccountAliasRequest](#) objet contenant le nom de l'alias.

Importations

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

Code

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

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

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

Voir l'[exemple complet](#) sur GitHub.

Liste des alias de compte

Pour obtenir la liste des alias de compte, le cas échéant, appelez la méthode `listAccountAliases` d'`AmazonIdentityManagementClient`.

Note

La méthode renvoyée [ListAccountAliasesResult](#) prend en charge les mêmes `getMarker` méthodes `getIsTruncated` et que les autres méthodes de AWS SDK for Java liste, mais un compte AWS peut avoir qu'un seul alias de compte.

importations

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

code

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

ListAccountAliasesResult response = iam.listAccountAliases();

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

voir l'[exemple complet](#) sur GitHub.

Suppression d'un alias de compte

Pour supprimer votre alias de compte, appelez la méthode `deleteAccountAlias` d'`AmazonIdentityManagementClient`. Lorsque vous supprimez un alias de compte, vous devez fournir son nom à l'aide d'un [DeleteAccountAliasRequest](#) objet.

importations

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

Code

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

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

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

Voir l'[exemple complet](#) sur GitHub.

En savoir plus

- [Votre identifiant deAWS compte et son alias](#) dans le guide deIAM l'utilisateur
- [CreateAccountAlias](#) dans la référence de l'API IAM
- [ListAccountAliases](#) dans la référence de l'API IAM
- [DeleteAccountAlias](#) dans la référence de l'API IAM

Utilisation des stratégies IAM

Création d'une politique

Pour créer une nouvelle stratégie, fournissez le nom de la stratégie et un document de stratégie au format JSON dans un [CreatePolicyRequest](#) vers le client `AmazonIdentityManagementClient.createPolicy` méthode.

Imports

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

Code

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

CreatePolicyRequest request = new CreatePolicyRequest()
```

```
.withPolicyName(policy_name)
.withPolicyDocument(POLICY_DOCUMENT);

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

Les documents de stratégie IAM sont des chaînes JSON avec un [syntaxe bien documentée](#). Voici un exemple qui fournit l'accès permettant d'adresser des demandes particulières à 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\"" +
    "    }" +
    "  ]" +
    "}";
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Obtention d'une stratégie

Pour récupérer une stratégie existante, appelez `getPolicy`, fournissant le nom ARN de la stratégie dans un [GetPolicyRequest](#) objet.

Importations

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

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

Code

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

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Attachement d'une stratégie de rôle

Vous pouvez attacher une stratégie à un IAM (http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html[role]) en appelant `attachRolePolicy`, en lui fournissant le nom du rôle et l'ARN de la stratégie dans un [AttachRolePolicyRequest](#).

Importations

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

Code

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

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

iam.attachRolePolicy(attach_request);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Affichage d'une liste de stratégies de rôle attachées

Répertoriez les stratégies attachées sur un rôle en appelant `iam.listAttachedRolePolicies` méthode. Cette méthode accepte un objet [ListAttachedRolePoliciesRequest](#) contenant le nom du rôle pour lequel répertorier les stratégies.

Appelez `getAttachedPolicies` sur l'objet [ListAttachedRolePoliciesResult](#) renvoyé pour obtenir la liste des stratégies attachées. Les résultats peuvent être tronqués. Si la méthode `getIsTruncated` de l'objet `ListAttachedRolePoliciesResult` renvoie `true`, appelez la méthode `setMarker` de l'objet `ListAttachedRolePoliciesRequest` et utilisez-la pour appeler à nouveau `listAttachedRolePolicies` afin d'obtenir le lot suivant de résultats.

Importations

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

Code

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

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

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

boolean done = false;

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

    matching_policies.addAll(
        response.getAttachedPolicies()
            .stream())
}
```

```
                .filter(p -> p.getPolicyName().equals(role_name))
                .collect(Collectors.toList());

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Détachement d'une stratégie de rôle

Pour détacher une stratégie d'un rôle, appelez le `detachRolePolicy`, en lui fournissant le nom du rôle et l'ARN de la stratégie dans un [DetachRolePolicyRequest](#).

Importations

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

Code

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

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [Présentation des politiques IAM](#) dans le IAM Guide de l'utilisateur .
- [AWS Référence de stratégie IAM](#) dans le IAM Guide de l'utilisateur .

- [CreatePolicy](#) dans la Référence d'API IAM
- [GetPolicy](#) dans la Référence d'API IAM
- [AttachRolePolicy](#) dans la Référence d'API IAM
- [ListAttachedRolePolicies](#) dans la Référence d'API IAM
- [DetachRolePolicy](#) dans la Référence d'API IAM

Utilisation des certificats de serveur IAM

Pour activer des connexions HTTPS vers votre site Web ou votre application sur AWS, vous avez besoin d'un protocole SSL/TLS serveur de certificats. Vous pouvez utiliser un certificat de serveur fourni par AWS Certificate Manager ou un gestionnaire que vous avez obtenu auprès d'un fournisseur externe.

Nous vous recommandons d'utiliser ACM pour mettre en service, gérer et déployer vos certificats de serveur. Avec ACM vous pouvez demander un certificat et le déployer vers votre AWS. Laissez ACM gérer le renouvellement des certificats pour vous. Les certificats fournis par ACM sont gratuits. Pour plus d'informations sur ACM, consultez le [Guide de l'utilisateur ACM](#).

Obtention d'un certificat de serveur

Vous pouvez récupérer un certificat de serveur en appelant le `getServerCertificate`, en lui transmettant un [GetServerCertificateRequest](#) avec le nom du certificat.

Importations

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

Code

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

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Liste des certificats de serveur

Pour répertorier vos certificats de serveur, appelez le `listServerCertificates` avec un [ListServerCertificatesRequest](#). Cette méthode renvoie un objet [ListServerCertificatesResult](#).

Appelez la méthode `getServerCertificateMetadataList` de l'objet `ListServerCertificateResult` renvoyé pour obtenir la liste des objets [ServerCertificateMetadata](#) permettant d'obtenir des informations sur chaque certificat.

Les résultats peuvent être tronqués. Si la méthode `getIsTruncated` de l'objet `ListServerCertificateResult` renvoie `true`, appelez la méthode `setMarker` de l'objet `ListServerCertificatesRequest` et utilisez-la pour appeler à nouveau `listServerCertificates` afin d'obtenir le lot suivant de résultats.

Importations

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

Code

```
final AmazonIdentityManagement iam =  
    AmazonIdentityManagementClientBuilder.defaultClient();  
  
boolean done = false;  
ListServerCertificatesRequest request =  
    new ListServerCertificatesRequest();  
  
while(!done) {  
  
    ListServerCertificatesResult response =  
        iam.listServerCertificates(request);  
  
    for(ServerCertificateMetadata metadata :
```



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

        request.setMarker(response.getMarker());

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Mise à jour d'un certificat de serveur

Vous pouvez mettre à jour le nom ou le chemin d'un certificat de serveur en appelant `le.updateServerCertificateMéthode`. Cette méthode prend un ensemble d'objets [UpdateServerCertificateRequest](#) avec le nom actuel du certificat de serveur, ainsi qu'un nouveau nom ou un nouveau chemin à utiliser.

Importations

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

Code

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

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Suppression d'un certificat de serveur

Pour supprimer un certificat de serveur, appelez le `deleteServerCertificate` avec un [DeleteServerCertificateRequest](#) contenant le nom du certificat.

Importations

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

Code

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

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [Utilisation des certificats de serveur](#) dans le IAM Guide de l'utilisateur
- [GetServerCertificate](#) dans la Référence d'API IAM
- [ListServerCertificates](#) dans la Référence d'API IAM
- [UpdateServerCertificate](#) dans la Référence d'API IAM
- [DeleteServerCertificate](#) dans la Référence d'API IAM
- [Guide de l'utilisateur ACM](#)

LambdaExemples avec utilisation de l'AWS SDK for Java

Cette section fournit des exemples de programmation d'Lambda à l'aide du kit AWS SDK for Java.

Note

Les exemples incluent uniquement le code nécessaire pour démontrer chaque technique. L'exemple de code complet [est disponible sur GitHub](#). À partir de là, vous pouvez télécharger un fichier source unique ou cloner le référentiel en local pour obtenir tous les exemples à générer et exécuter.

Rubriques

- [Appel, affichage et suppression LambdaFonctions](#)

Appel, affichage et suppression LambdaFonctions

Cette section fournit des exemples de programmation à l'aide du `LambdaClient` de service en utilisant le AWS SDK for Java. Pour savoir comment créer un `LambdaFunction`, consultez [Comment créer AWS Lambda fonctions](#).

Rubriques

- [Appeler une fonction](#)
- [Répertoire des fonctions](#)
- [Supprimer une fonction](#)

Appeler une fonction

Vous pouvez appeler un `Lambda` en créant une fonction `AWSLambda` objet et invoquant son `invoke` méthode. Créez un objet `InvokeRequest` pour spécifier des informations supplémentaires telles que le nom de la fonction et la charge utile à transmettre à la fonction Lambda. Les noms de fonctions apparaissent sous la forme `arn:aws:lambda:us-east-1:5556330391:function:HelloFunction`. Vous pouvez récupérer la valeur en observant la fonction dans la AWS Management Console.

Pour transmettre des données de charge utile à une fonction, appelez l'`InvokeRequest` objet avec `payload` et spécifiez une chaîne au format JSON, comme illustré dans l'exemple de code suivant.

Importations

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.lambda.AWSLambda;
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;
import com.amazonaws.services.lambda.model.InvokeRequest;
import com.amazonaws.services.lambda.model.InvokeResult;
import com.amazonaws.services.lambda.model.ServiceException;

import java.nio.charset.StandardCharsets;
```

Code

L'exemple de code suivant illustre comment appeler une fonction 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());
```

Consultez l'exemple complet sur [GitHub](#).

Répertoire des fonctions

Construisez un objet [AWSLambda](#) et invoquez sa méthode `listFunctions`. Cette méthode renvoie un objet [ListFunctionsResult](#). Vous pouvez appeler la méthode `getFunctions` de cet objet pour renvoyer une liste des objets [FunctionConfiguration](#). Parcourez la liste pour récupérer des informations sur les fonctions. Par exemple, l'exemple de code Java ci-dessous illustre comment obtenir le nom de chaque fonction.

Importations

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.lambda.AWSLambda;
import com.amazonaws.services.lambda.AWSLambdaClientBuilder;
import com.amazonaws.services.lambda.model.FunctionConfiguration;
import com.amazonaws.services.lambda.model.ListFunctionsResult;
import com.amazonaws.services.lambda.model.ServiceException;
import java.util.Iterator;
import java.util.List;
```

Code

L'exemple de code Java suivant illustre comment récupérer une liste de noms de fonctions.

```
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);
    }
```

Consultez l'exemple complet sur [GitHub](#).

Supprimer une fonction

Construisez un objet [AWSLambda](#) et invoquez sa méthode `deleteFunction`. Créez un objet [DeleteFunctionRequest](#) et transmettez-le à la méthode `deleteFunction`. Cet objet contient des informations telles que le nom de la fonction à supprimer. Les noms de fonctions apparaissent sous la forme `arn:aws:lambda:us-east-1:5556330391:function:HelloFunction`. Vous pouvez récupérer la valeur en observant la fonction dans la [AWS Management Console](#).

Importations

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

Code

Le code Java suivant illustre comment supprimer un 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);
}
```

```
}
```

Consultez l'exemple complet sur [GitHub](#).

Amazon PinpointExemples avec utilisation de l'AWS SDK for Java

Cette section fournit des exemples de programmation d'[Amazon Pinpoint](#) à l'aide du kit [AWS SDK for Java](#).

Note

Les exemples incluent uniquement le code nécessaire pour démontrer chaque technique. L'exemple de code complet [est disponible sur GitHub](#). À partir de là, vous pouvez télécharger un fichier source unique ou cloner le référentiel en local pour obtenir tous les exemples à générer et exécuter.

Rubriques

- [Création et suppression d'applications dansAmazon Pinpoint](#)
- [Création de points de terminaison dansAmazon Pinpoint](#)
- [Création de segments dansAmazon Pinpoint](#)
- [Création de campagnes dansAmazon Pinpoint](#)
- [Mise à jour des canaux dansAmazon Pinpoint](#)

Création et suppression d'applications dansAmazon Pinpoint

Une application est un projet Amazon Pinpoint dans lequel vous définissez le public ciblé pour une application distincte, et vous impliquez ce public avec des messages sur mesure. Les exemples de cette page montrent comment créer une application ou comment supprimer une application existante.

Création d'une application

Créez une nouvelle application dansAmazon Pinpointen fournissant un nom d'application au[CreateAppRequest](#), puis en transmettant cet objet à l'objet `AmazonPinpointClientcreateApp`méthode.

Importations

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

Code

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Suppression d'une application

Pour supprimer une application, appelez `deleteApp` de la demande d'avec un [DeleteAppRequest](#) objet défini avec le nom de l'application à supprimer.

Importations

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

Code

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

pinpoint.deleteApp(deleteRequest);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [Applications](#) dans le Amazon Pinpoint API Reference
- [App](#) dans le Amazon Pinpoint API Reference

Création de points de terminaison dans Amazon Pinpoint

Un point de terminaison identifie de façon unique l'appareil d'un utilisateur auquel vous pouvez envoyer des notifications push avec Amazon Pinpoint. Si votre application est activée avec la prise en charge d'Amazon Pinpoint, elle inscrit automatiquement le point de terminaison de l'utilisateur auprès d'Amazon Pinpoint lorsqu'il ouvre l'application pour la première fois. L'exemple suivant montre comment ajouter un nouveau point de terminaison par programmation.

Création d'un point de terminaison

Pour créer un point de terminaison dans Amazon Pinpoint, fournissez les données du point de terminaison dans un objet [EndpointRequest](#).

Importations

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.UpdateEndpointRequest;
import com.amazonaws.services.pinpoint.model.UpdateEndpointResult;
import com.amazonaws.services.pinpoint.model.EndpointDemographic;
import com.amazonaws.services.pinpoint.model.EndpointLocation;
import com.amazonaws.services.pinpoint.model.EndpointRequest;
import com.amazonaws.services.pinpoint.model.EndpointResponse;
import com.amazonaws.services.pinpoint.model.EndpointUser;
import com.amazonaws.services.pinpoint.model.GetEndpointRequest;
import com.amazonaws.services.pinpoint.model.GetEndpointResult;
```

Code

```
HashMap<String, List<String>> customAttributes = new HashMap<>();
List<String> favoriteTeams = new ArrayList<>();
favoriteTeams.add("Lakers");
favoriteTeams.add("Warriors");
customAttributes.put("team", favoriteTeams);

EndpointDemographic demographic = new EndpointDemographic()
    .withAppVersion("1.0")
    .withMake("apple")
    .withModel("iPhone")
    .withModelVersion("7")
    .withPlatform("ios")
```

```
.withPlatformVersion("10.1.1")
.withTimezone("America/Los_Angeles");

EndpointLocation location = new EndpointLocation()
    .withCity("Los Angeles")
    .withCountry("US")
    .withLatitude(34.0)
    .withLongitude(-118.2)
    .withPostalCode("90068")
    .withRegion("CA");

Map<String,Double> metrics = new HashMap<>();
metrics.put("health", 100.00);
metrics.put("luck", 75.00);

EndpointUser user = new EndpointUser()
    .withUserId(UUID.randomUUID().toString());

DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted "Z" to
    indicate UTC, no timezone offset
String nowAsISO = df.format(new Date());

EndpointRequest endpointRequest = new EndpointRequest()
    .withAddress(UUID.randomUUID().toString())
    .withAttributes(customAttributes)
    .withChannelType("APNS")
    .withDemographic(demographic)
    .withEffectiveDate(nowAsISO)
    .withLocation(location)
    .withMetrics(metrics)
    .withOptOut("NONE")
    .withRequestId(UUID.randomUUID().toString())
    .withUser(user);
```

Créez ensuite un [UpdateEndpointRequest](#) avec l'objet `EndpointRequest` l'objet. Enfin, passez le `UpdateEndpointRequest` à l'objet `AmazonPinpointClient` updateEndpointRequest méthode.

Code

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [Ajout d'un point de](#) dans le Amazon Pinpoint Manuel du développeur
- [Point de terminaison](#) dans le Amazon Pinpoint API Reference

Création de segments dans Amazon Pinpoint

Un segment d'utilisateurs est un sous-ensemble d'utilisateurs qui présentent des caractéristiques communes. Il peut s'agir de la date à laquelle les utilisateurs ont ouvert votre application pour la dernière fois ou du type d'appareil qu'ils utilisent. L'exemple suivant montre comment définir un segment d'utilisateurs.

Créer un segment

Pour créer un segment dans Amazon Pinpoint, vous devez définir les dimensions du segment dans un objet [SegmentDimensions](#).

Importations

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateSegmentRequest;
import com.amazonaws.services.pinpoint.model.CreateSegmentResult;
import com.amazonaws.services.pinpoint.model.AttributeDimension;
import com.amazonaws.services.pinpoint.model.AttributeType;
import com.amazonaws.services.pinpoint.model.RecencyDimension;
import com.amazonaws.services.pinpoint.model.SegmentBehaviors;
import com.amazonaws.services.pinpoint.model.SegmentDemographics;
import com.amazonaws.services.pinpoint.model.SegmentDimensions;
import com.amazonaws.services.pinpoint.model.SegmentLocation;
import com.amazonaws.services.pinpoint.model.SegmentResponse;
import com.amazonaws.services.pinpoint.model.WriteSegmentRequest;
```

Code

```
Pinpoint pinpoint =
    AmazonPinpointClientBuilder.standard().withRegion(Regions.US_EAST_1).build();
Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
segmentAttributes.put("Team", new
    AttributeDimension().withAttributeType(AttributeType.INCLUSIVE).withValues("Lakers"));

SegmentBehaviors segmentBehaviors = new SegmentBehaviors();
SegmentDemographics segmentDemographics = new SegmentDemographics();
SegmentLocation segmentLocation = new SegmentLocation();

RecencyDimension recencyDimension = new RecencyDimension();
recencyDimension.withDuration("DAY_30").withRecencyType("ACTIVE");
segmentBehaviors.setRecency(recencyDimension);

SegmentDimensions dimensions = new SegmentDimensions()
    .withAttributes(segmentAttributes)
    .withBehavior(segmentBehaviors)
    .withDemographic(segmentDemographics)
    .withLocation(segmentLocation);
```

Définissez ensuite l'objet [SegmentDimensions](#) dans un objet [WriteSegmentRequest](#), qui sera à son tour utilisé pour créer un objet [CreateSegmentRequest](#). Ensuite, passez le `CreateSegmentRequest` à l'objet `AmazonPinpointClient` `createSegment` méthode.

Code

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

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

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

Veuillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [Amazon PinpointSegments](#) dans le `Amazon Pinpoint` Guide de l'utilisateur
- [Création de segments](#) dans le `Amazon Pinpoint` Manuel du développeur

- [Segments](#) dans le Amazon Pinpoint API Reference
- [Segment](#) dans le Amazon Pinpoint API Reference

Création de campagnes dans Amazon Pinpoint

Les campagnes vous permettent de renforcer l'implication des utilisateurs vis-à-vis de votre application. Vous pouvez créer une campagne pour toucher un segment particulier d'utilisateurs à l'aide de messages sur mesure ou de promotions spéciales. Cet exemple montre comment créer une campagne standard qui envoie une notification push personnalisée à un segment déterminé.

Création d'une campagne

Avant de créer une nouvelle campagne, vous devez définir un [calendrier](#) et un [message](#) et définir ces valeurs dans un [WriteCampaignRequest](#) objet.

Importations

```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateCampaignRequest;
import com.amazonaws.services.pinpoint.model.CreateCampaignResult;
import com.amazonaws.services.pinpoint.model.Action;
import com.amazonaws.services.pinpoint.model.CampaignResponse;
import com.amazonaws.services.pinpoint.model.Message;
import com.amazonaws.services.pinpoint.model.MessageConfiguration;
import com.amazonaws.services.pinpoint.model.Schedule;
import com.amazonaws.services.pinpoint.model.WriteCampaignRequest;
```

Code

```
Schedule schedule = new Schedule()
    .withStartTime("IMMEDIATE");

Message defaultMessage = new Message()
    .withAction(Action.OPEN_APP)
    .withBody("My message body.")
    .withTitle("My message title.");

MessageConfiguration messageConfiguration = new MessageConfiguration()
    .withDefaultMessage(defaultMessage);
```

```
WriteCampaignRequest request = new WriteCampaignRequest()
    .withDescription("My description.")
    .withSchedule(schedule)
    .withSegmentId(segmentId)
    .withName("MyCampaign")
    .withMessageConfiguration(messageConfiguration);
```

Créez ensuite une nouvelle campagne en Amazon Pinpoint fournissant la configuration [WriteCampaignRequest](#) de la campagne à un [CreateCampaignRequest](#) objet. Enfin, passez l'[CreateCampaignRequest](#) objet à [AmazonPinpointClient](#) la `createCampaign` méthode s.

Code

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

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

Voir l'[exemple complet](#) sur GitHub.

En savoir plus

- [Amazon Pinpoint Campagnes](#) dans le guide de Amazon Pinpoint l'utilisateur
- [Création de campagnes](#) dans le guide du Amazon Pinpoint développeur
- [Campagnes](#) dans l'Amazon Pinpoint API Reference
- [Campagne](#) dans l'Amazon Pinpoint API Reference
- [Activités de campagne](#) dans l'Amazon Pinpoint API Reference
- [Versions de campagne](#) dans le guide de référence de Amazon Pinpoint l'API
- [Version de la campagne](#) dans la référence de Amazon Pinpoint l'API

Mise à jour des canaux dans Amazon Pinpoint

Un canal définit les types de plateformes auxquelles vous pouvez envoyer des messages. Cet exemple montre comment utiliser le canal du service APN pour envoyer un message.

Mise à jour d'un canal

Pour activer un canal dans Amazon Pinpoint, vous devez fournir l'ID d'une application et un objet de demande correspondant au type de canal que vous souhaitez mettre à jour. Cet exemple

met à jour le canal du service APN, ce qui nécessite l'objet [APNSChannelRequest](#). Définissez ces éléments dans le [UpdateApnsChannelRequest](#) et transmettez cet objet au service du service `AmazonPinpointClient.updateApnsChannel` méthode.

Importations

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

Code

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

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

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [Amazon Pinpoint Chaînes](#) dans le Amazon Pinpoint Guide de l'utilisateur
- [Canal ADM](#) dans le Amazon Pinpoint API Reference
- [Canal du service APN](#) dans le Amazon Pinpoint API Reference
- [Canal de l'établissement \(sandbox\)](#) dans le Amazon Pinpoint API Reference
- [Canal VoIP du service APN](#) dans le Amazon Pinpoint API Reference
- [Canal du service de test \(sandbox\) VoIP](#) dans le Amazon Pinpoint API Reference
- [Canal du service Baidu](#) dans le Amazon Pinpoint API Reference
- [Canal des e-mails](#) dans le Amazon Pinpoint API Reference

- [Canal GCM](#) dans le Amazon Pinpoint API Reference
- [Canal des SMS](#) dans le Amazon Pinpoint API Reference

Amazon S3 Exemples avec utilisation de l'AWS SDK for Java

Cette section fournit des exemples de programmation d'[Amazon S3](#) à l'aide du kit [AWS SDK for Java](#).

Note

Les exemples incluent uniquement le code nécessaire pour démontrer chaque technique. L'exemple de code complet [est disponible sur GitHub](#). À partir de là, vous pouvez télécharger un fichier source unique ou cloner le référentiel en local pour obtenir tous les exemples à générer et exécuter.

Rubriques

- [Création, affichage et suppression Amazon S3 Compartiments](#)
- [Exécution d'opérations sur Amazon S3 des objets](#)
- [Gestion d'Amazon S3 Autorisations d'accès des compartiments et objets](#)
- [Gestion de l'accès à Amazon S3 Utilisation de stratégies de compartiment](#)
- [A l'aide de TransferManager pour Amazon S3 Opérations](#)
- [Configuration d'un Amazon S3 Bucket en tant que site web](#)
- [Utiliser Amazon S3 Chiffrement côté client](#)

Création, affichage et suppression Amazon S3 Compartiments

Chaque objet (fichier) d'Amazon S3 doit résider dans un compartiment, qui représente un ensemble (conteneur) d'objets. Chaque compartiment est identifié par une clé (nom) qui doit être unique. Pour de plus amples informations sur les compartiments et leur configuration, veuillez consulter [Utilisation d'Amazon S3 Compartiments](#) dans le Amazon Simple Storage Service Guide de l'utilisateur .

Note

Bonne pratique

Nous vous recommandons d'activer la règle de cycle de vie [AbortIncompleteMultipartUpload](#) sur vos compartiments Amazon S3.

Cette règle indique à Amazon S3 d'interrompre les chargements partitionnés qui ne sont pas terminés après un nombre de jours prédéfini à compter de leur lancement. Lorsque le délai défini est dépassé, Amazon S3 interrompt le chargement, puis supprime les données du chargement inachevé.

Pour de plus amples informations, veuillez consulter [Configuration du cycle de vie pour un compartiment avec gestion des versions](#) dans le [Amazon S3 Guide de l'utilisateur](#) .

Note

Ces exemples de code supposent que vous comprenez le contenu de [Utilisation de AWS SDK for Java](#) et ont configuré la valeur par défaut AWS informations d'identification utilisant les informations de [Configuration AWS Informations d'identification et région pour le développement](#).

Création d'un compartiment

Utilisation du client `AmazonS3createBucket` Méthode. Le nouveau [compartiment](#) est renvoyé. La méthode `createBucket` déclenche une exception si le compartiment existe déjà.

Note

Pour vérifier si un compartiment existe déjà avant de tenter d'en créer un avec le même nom, appelez la méthode `doesBucketExist`. Cette méthode renvoie `true` si le compartiment existe et `false` sinon.

Importations

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

```
import java.util.List;
```

Code

```
if (s3.doesBucketExistV2(bucket_name)) {
    System.out.format("Bucket %s already exists.\n", bucket_name);
    b = getBucket(bucket_name);
} else {
    try {
        b = s3.createBucket(bucket_name);
    } catch (AmazonS3Exception e) {
        System.err.println(e.getErrorMessage());
    }
}
return b;
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Etablir une liste des compartiments

Utilisation du client AmazonS3 `listBucket` Méthode. En cas de réussite, une liste de [compartiments](#) est renvoyée.

Importations

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

import java.util.List;
```

Code

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Supprimer un compartiment

Pour pouvoir supprimer un compartiment Amazon S3, vous devez vous assurer que le compartiment est vide, sinon une erreur se produit. S'il s'agit d'un [compartiment avec gestion des versions](#), vous devez également supprimer tous les objets versionnés associés à celui-ci.

Note

L'[exemple complet](#) comprend chacune de ces étapes dans l'ordre, en fournissant une solution complète pour supprimer un compartiment Amazon S3 et son contenu.

Rubriques

- [Suppression des objets d'un compartiment sans gestion des versions avant sa suppression](#)
- [Suppression des objets d'un compartiment avec gestion des versions avant sa suppression](#)
- [Suppression d'un compartiment vide](#)

Suppression des objets d'un compartiment sans gestion des versions avant sa suppression

Utilisation du client `AmazonS3listObjects` pour extraire la liste des objets et `deleteObject` pour supprimer chacun d'eux.

Importations

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

import java.util.Iterator;
```

Code

```
System.out.println(" - removing objects from bucket");
ObjectListing object_listing = s3.listObjects(bucket_name);
while (true) {
    for (Iterator<?> iterator =
        object_listing.getObjectSummaries().iterator();
```

```
        iterator.hasNext(); ) {
        S3ObjectSummary summary = (S3ObjectSummary) iterator.next();
        s3.deleteObject(bucket_name, summary.getKey());
    }

    // more object_listing to retrieve?
    if (object_listing.isTruncated()) {
        object_listing = s3.listNextBatchOfObjects(object_listing);
    } else {
        break;
    }
}
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Suppression des objets d'un compartiment avec gestion des versions avant sa suppression

Si vous utilisez un [compartiment avec gestion des versions](#), vous devez également supprimer toutes les versions stockées des objets du compartiment pour que le compartiment puisse être supprimé.

En appliquant un modèle similaire à celui utilisé lors de la suppression d'objets d'un compartiment, supprimez les objets versionnés à l'aide du `listVersions` pour répertorier tous les objets versionnés, puis `deleteVersion` pour supprimer chacun d'eux.

Importations

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

import java.util.Iterator;
```

Code

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

```
S3VersionSummary vs = (S3VersionSummary) iterator.next();
s3.deleteVersion(
    bucket_name, vs.getKey(), vs.getVersionId());
}

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Suppression d'un compartiment vide

Une fois que vous avez supprimé les objets d'un compartiment (y compris tous les objets versionnés), vous pouvez supprimer le compartiment proprement dit à l'aide de `deleteBucket` Méthode.

Importations

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

import java.util.Iterator;
```

Code

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Exécution d'opérations sur Amazon S3 des objets

Un objet Amazon S3 représente un fichier ou un ensemble de données. Chaque objet doit résider dans un [compartiment](#).

Note

Ces exemples de code supposent que vous comprenez le contenu de la section [Utilisation du AWS SDK for Java et que vous avez configuré les](#) AWS informations d'identification par défaut à l'aide des informations fournies dans [Configurer les informations AWS d'identification et la région pour le développement](#).

Rubriques

- [Chargement d'un objet](#)
- [Affichage de la liste des objets](#)
- [Téléchargement d'un objet](#)
- [Copie et déplacement d'objets, ou attribution d'un nouveau nom aux objets](#)
- [Supprimer un objet](#)
- [Suppression simultanée de plusieurs objets](#)

Chargement d'un objet

Utilisez la `putObject` méthode du client AmazonS3, en fournissant un nom de compartiment, un nom de clé et un fichier à charger. Le compartiment doit exister, sans quoi une erreur est générée.

Importations

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

Code

```
System.out.format("Uploading %s to S3 bucket %s...\n", file_path, bucket_name);
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.putObject(bucket_name, key_name, new File(file_path));
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

```
}
```

Voir l'[exemple complet](#) surGitHub.

Affichage de la liste des objets

Pour obtenir la liste des objets contenus dans un bucket, utilisez la `listObjects` méthode du client `AmazonS3` en fournissant le nom du bucket.

La `listObjects` méthode renvoie un [ObjectListing](#) objet qui fournit des informations sur les objets du bucket. Pour répertorier les noms d'objets (clés), utilisez la `getObjectSummaries` méthode pour obtenir une liste d'`ObjectSummary` objets [S3](#), dont chacun représente un seul objet dans le bucket. Ensuite, appelez sa méthode `getKey` pour récupérer le nom de l'objet.

Importations

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

Code

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

Voir l'[exemple complet](#) surGitHub.

Téléchargement d'un objet

Utilisez la `getObject` méthode du client `AmazonS3` en lui transmettant le nom du bucket et de l'objet à télécharger. En cas de réussite, la méthode renvoie un objet [S3Object](#). Le compartiment et la clé d'objet spécifiés doivent exister, sans quoi une erreur est générée.

Vous pouvez obtenir le contenu de l'objet en appelant `getObjectContent` sur l'objet `S3Object`. Cela renvoie un [S3 ObjectInputStream](#) qui se comporte comme un `InputStream` objet Java standard.

L'exemple suivant télécharge un objet à partir de S3 et enregistre son contenu dans un fichier (en utilisant le même nom que la clé de l'objet).

Importations

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

import java.io.File;
```

Code

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



```
}
```

Voir l'[exemple complet](#) surGitHub.

Copie et déplacement d'objets, ou attribution d'un nouveau nom aux objets

Il se peut que vous souhaitiez copier un compartiment à un autre grâce à la méthode du `copyObject` client AmazonS3. Elle récupère le nom du compartiment d'où l'objet est copié, l'objet à copier et le nom du compartiment de destination.

Importations

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Code

```
try {
    s3.copyObject(from_bucket, object_key, to_bucket, object_key);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Done!");
```

Voir l'[exemple complet](#) surGitHub.

Note

Vous pouvez utiliser `copyObject` avec [deleteObject](#) pour déplacer ou renommer un objet, en copiant d'abord l'objet avec un nouveau nom (vous pouvez utiliser le même compartiment comme source et comme destination), puis en supprimant l'objet de son ancien emplacement.

Supprimer un objet

Utilisez la `deleteObject` méthode du client AmazonS3 en lui transmettant le nom du bucket et de l'objet à supprimer. Le compartiment et la clé d'objet spécifiés doivent exister, sans quoi une erreur est générée.

Importations

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.deleteObject(bucket_name, object_key);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Voir l'[exemple complet](#) sur GitHub.

Suppression simultanée de plusieurs objets

À l'aide de la `deleteObjects` méthode du client `AmazonS3`, vous pouvez supprimer plusieurs objets du même bucket en transmettant leurs noms à la méthode `link` : `sdk-for-java DeleteObjectsRequest /v1/reference/com/amazonaws/services/s3/model/ .html`.

Importations

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

Code

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

```
}
```

Voir l'[exemple complet](#) sur GitHub.

Gestion d'Amazon S3 Autorisations d'accès des compartiments et objets

Vous pouvez utiliser des listes de contrôle d'accès (ACL) pour les compartiments et les objets Amazon S3 afin de bénéficier d'un contrôle précis sur vos ressources Amazon S3.

Note

Ces exemples de code supposent que vous comprenez le contenu de [Utilisation de AWS SDK for Java](#) et ont configuré la valeur par défaut AWS informations d'identification utilisant les informations de [Configuration AWS Informations d'identification et région pour le développement](#).

Obtention de la liste de contrôle d'accès pour un compartiment

Pour obtenir la liste ACL actuelle pour un compartiment, appelez le code Amazon S3 `getBucketAcl`, en lui transmettant le nom de compartiment pour interroger. Cette méthode renvoie un objet [AccessControlList](#). Pour obtenir chaque autorisation d'accès de la liste, appelez sa méthode `getGrantsAsList`, qui renvoie une liste Java standard d'objets [Grant](#).

Importations

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

Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    AccessControlList acl = s3.getBucketAcl(bucket_name);
```

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

Veuillez consulter l'[exemple complet](#) sur GitHub.

Définition de la liste de contrôle d'accès pour un compartiment

Pour ajouter ou modifier des autorisations dans une liste ACL pour un compartiment, appelez `setBucketAcl` méthode. Cette méthode prend un objet [AccessControlList](#) qui contient une liste de bénéficiaires et de niveaux d'accès à définir.

Importations

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

Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    // get the current ACL
    AccessControlList acl = s3.getBucketAcl(bucket_name);
    // set access for the grantee
    EmailAddressGrantee grantee = new EmailAddressGrantee(email);
    Permission permission = Permission.valueOf(access);
    acl.grantPermission(grantee, permission);
    s3.setBucketAcl(bucket_name, acl);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
```

}

Note

Vous pouvez directement fournir l'identifiant unique du bénéficiaire à l'aide de la classe [Grantee](#), ou utiliser la classe [EmailAddressGrantee](#) pour définir le bénéficiaire par adresse e-mail, comme nous l'avons fait ici.

Veillez consulter l'[exemple complet](#) sur GitHub.

Obtention de la liste de contrôle d'accès pour un objet

Pour obtenir la liste ACL actuelle pour un objet, appelez le code `Amazon S3.getObjectAcl`, en lui transmettant le nom de compartiment et le nom de l'objet pour interroger. Comme `getBucketAcl`, cette méthode renvoie un objet [AccessControlList](#) que vous pouvez utiliser pour examiner chaque objet [Grant](#).

Importations

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

Code

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Définition de la liste de contrôle d'accès pour un objet

Pour ajouter ou modifier des autorisations dans une liste ACL pour un objet, appelez le code Amazon S3 `setObjectAcl` méthode. Cette méthode prend un objet [AccessControlList](#) qui contient une liste de bénéficiaires et de niveaux d'accès à définir.

Importations

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

Code

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

Note

Vous pouvez directement fournir l'identifiant unique du bénéficiaire à l'aide de la classe [Grantee](#), ou utiliser la classe [EmailAddressGrantee](#) pour définir le bénéficiaire par adresse e-mail, comme nous l'avons fait ici.

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [GET Bucket acl](#) dans le Amazon S3 API Reference
- [PUT Bucket acl](#) dans le Amazon S3 API Reference
- [GET Object acl](#) dans le Amazon S3 API Reference
- [PUT Object acl](#) dans le Amazon S3 API Reference

Gestion de l'accès à Amazon S3 Utilisation de stratégies de compartiment

Vous pouvez définir, obtenir ou supprimer une stratégie de compartiment pour gérer l'accès à vos compartiments Amazon S3.

Définition d'une stratégie de compartiment

Vous pouvez définir la stratégie de compartiment pour un compartiment S3 :

- Appel du client `AmazonS3.setBucketPolicy` et lui fournir un [SetBucketPolicyRequest](#)
- En définissant la stratégie directement à l'aide de la surcharge `setBucketPolicy` qui prend un nom de compartiment et un texte de stratégie (au format JSON)

Importations

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
```

Code

```
s3.setBucketPolicy(bucket_name, policy_text);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Utilisation de la classe Policy pour générer ou valider une stratégie

Lorsque vous fournissez une stratégie de compartiment à `setBucketPolicy`, vous pouvez effectuer les actions suivantes :

- Spécifier la stratégie directement sous la forme d'une chaîne de texte au format JSON
- Créer la stratégie à l'aide de la classe [Policy](#)

En utilisant la classe `Policy`, vous n'avez pas à vous soucier de formater correctement votre chaîne de texte. Pour obtenir le texte de stratégie JSON à partir de la classe `Policy`, utilisez sa méthode `toJson`.

Importations

```
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

Code

```
new Statement(Statement.Effect.Allow)
    .withPrincipals(Principal.AllUsers)
    .withActions(S3Actions.GetObject)
    .withResources(new Resource(
        "{region-arn}s3::" + bucket_name + "/*"));
return bucket_policy.toJson();
```

La classe `Policy` fournit également une méthode `fromJson` qui peut tenter de créer une stratégie à l'aide d'une chaîne JSON transmise. La méthode valide cette dernière pour s'assurer que le texte peut être transformé en une structure de stratégie valide et échoue avec une exception `IllegalArgumentException` si le texte de la stratégie n'est pas valide.

```
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());
}
```


Vous pouvez utiliser cette technique pour prévalider une stratégie que vous lisez à partir d'un fichier ou avec tout autre moyen.

Veillez consulter l'[exemple complet](#) sur GitHub.

Obtention d'une stratégie de compartiment

Pour récupérer la stratégie pour un Amazon S3, appelez le client `AmazonS3getBucketPolicy` en lui transmettant le nom du compartiment à partir lequel récupérer la stratégie.

Importations

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

Code

```
try {
    BucketPolicy bucket_policy = s3.getBucketPolicy(bucket_name);
    policy_text = bucket_policy.getPolicyText();
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Si le compartiment nommé n'existe pas, si vous n'y avez pas accès ou s'il n'a pas de stratégie de compartiment, une exception `AmazonServiceException` est levée.

Veillez consulter l'[exemple complet](#) sur GitHub.

Suppression d'une stratégie de compartiment

Pour supprimer une stratégie de compartiment, appelez la méthode du client `AmazonS3deleteBucketPolicy` en lui fournissant le nom du compartiment.

Importations

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
```

Code

```
try {
    s3.deleteBucketPolicy(bucket_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Cette méthode aboutit même si le compartiment ne comporte pas encore de stratégie. Si vous spécifiez un nom de compartiment qui n'existe pas ou si vous n'avez pas accès au compartiment, une exception `AmazonServiceException` est levée.

Veillez consulter l'[exemple complet](#) sur GitHub.

Plus d'informations

- [Présentation du langage d'accès policy](#) dans le [Amazon Simple Storage Service Guide](#) de l'utilisateur
- [Exemples de stratégie de compartiment](#) dans le [Amazon Simple Storage Service Guide](#) de l'utilisateur

A l'aide de `TransferManager` pour Amazon S3 Opérations

Vous pouvez utiliser la classe `TransferManager` du kit AWS SDK for Java pour transférer des fichiers de manière fiable de l'environnement local vers Amazon S3 et pour copier des objets d'un emplacement S3 vers un autre. `TransferManager` peut obtenir la progression d'un transfert, et suspendre ou reprendre les chargements et les téléchargements.

Note

Bonne pratique

Nous vous recommandons d'activer la règle de cycle de vie [AbortIncompleteMultipartUpload](#) sur vos compartiments Amazon S3.

Cette règle indique à Amazon S3 d'interrompre les chargements partitionnés qui ne sont pas terminés après un nombre de jours prédéfini à compter de leur lancement. Lorsque le délai

défini est dépassé, Amazon S3 interrompt le chargement, puis supprime les données du chargement inachevé.

Pour de plus amples informations, veuillez consulter [Configuration du cycle de vie pour un compartiment avec gestion des versions](#) dans le [Amazon S3 Guide de l'utilisateur](#).

Note

Ces exemples de code supposent que vous comprenez le contenu de [Utilisation de AWS SDK for Java](#) et ont configuré la valeur par défaut AWS informations d'identification utilisant les informations de [Configuration AWS Informations d'identification et région pour le développement](#).

Chargement des fichiers et des répertoires

TransferManager peut charger des fichiers, des listes de fichiers et des répertoires sur tout Amazon S3 seaux que vous avez [précédemment créée](#).

Rubriques

- [Chargement d'un seul fichier](#)
- [Chargement d'une liste de fichiers](#)
- [Charger un répertoire](#)

Chargement d'un seul fichier

Appelez TransferManager's `upload`, fournissant une méthode Amazon S3 Nom du compartiment, nom de clé (objet) et Java standard [Fichier](#) qui représente le fichier à charger.

Importations

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

import java.io.File;
```

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

Code

```
File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Upload xfer = xfer_mgr.upload(bucket_name, key_name, f);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

La méthode `upload` renvoie immédiatement un résultat, en fournissant un objet `Upload` à utiliser pour vérifier l'état du transfert ou attendre qu'il se termine.

Voir [Attente de la fin d'un transfert](#) Informations supplémentaires sur l'utilisation de `waitForCompletion` pour effectuer un transfert avec succès avant d'appeler `TransferManager.shutdownNow` Méthode. En attendant que le transfert se termine, vous pouvez interroger ou écouter les mises à jour relatives à son état et à sa progression. Pour plus d'informations, consultez [Obtention de l'état et de la progression du transfert](#).

Veillez consulter l'[exemple complet](#) sur GitHub.

Chargement d'une liste de fichiers

Pour charger plusieurs fichiers en une seule opération, appelez le `TransferManager.uploadFileList`, en fournissant les éléments suivants :

- Un nom de compartiment Amazon S3
- Un préfixe de clé à ajouter devant les noms des objets créés (le chemin au sein du compartiment dans lequel placer les objets)
- Un objet [File](#) qui représente le répertoire relatif à partir duquel créer les chemins de fichier
- Un objet [List](#) contenant un ensemble d'objets [File](#) à charger

Importations

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

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

Code

```
ArrayList<File> files = new ArrayList<File>();
for (String path : file_paths) {
    files.add(new File(path));
}

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

Voir [Attente de la fin d'un transfert](#) Informations supplémentaires sur l'utilisation de `waitForCompletion` pour effectuer un transfert avec succès avant d'appeler `TransferManager.shutdownNow` Méthode. En attendant que le transfert se termine, vous pouvez interroger ou écouter les mises à jour relatives à son état et à sa progression. Pour plus d'informations, consultez [Obtention de l'état et de la progression du transfert](#).

L'objet [MultipleFileUpload](#) renvoyé par `uploadFileList` peut être utilisé pour interroger l'état ou la progression du transfert. Pour plus d'informations, consultez [Interrogation de la progression en cours d'un transfert](#) et [Obtention de la progression d'un transfert avec un ProgressListener](#).

Vous pouvez aussi utiliser la méthode `MultipleFileUpload` de `getSubTransfers` pour obtenir les objets `Upload` individuels de chaque fichier transféré. Pour plus d'informations, consultez [Obtention de la progression des sous-transferts](#).

Veillez consulter l'[exemple complet](#) sur GitHub.

Charger un répertoire

Vous pouvez utiliser `uploadDirectory` pour charger l'intégralité d'un répertoire de fichiers, avec la possibilité de copier les fichiers dans les sous-répertoires de façon récursive. Vous fournissez un nom de compartiment Amazon S3, un préfixe de clé S3, un objet [File](#) représentant le répertoire local à copier et une valeur `boolean` indiquant si vous souhaitez copier les sous-répertoires de façon récursive (`true` ou `false`).

Importations

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

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

Code

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

Voir [Attente de la fin d'un transfert](#) Informations supplémentaires sur l'utilisation de `waitForCompletion` pour effectuer un transfert avec succès avant d'appeler `TransferManager.shutdownNow` Méthode. En attendant que le transfert se termine, vous pouvez interroger ou écouter les mises à jour relatives à son état et à sa progression. Pour plus d'informations, consultez [Obtention de l'état et de la progression du transfert](#).

L'objet [MultipleFileUpload](#) renvoyé par `uploadFileList` peut être utilisé pour interroger l'état ou la progression du transfert. Pour plus d'informations, consultez [Interrogation de la progression en cours d'un transfert](#) et [Obtention de la progression d'un transfert avec un ProgressListener](#).

Vous pouvez aussi utiliser la méthode `MultipleFileUpload` de `getSubTransfers` pour obtenir les objets `Upload` individuels de chaque fichier transféré. Pour plus d'informations, consultez [Obtention de la progression des sous-transferts](#).

Veillez consulter l'[exemple complet](#) sur GitHub.

Téléchargement de fichiers ou de répertoires

Utilisation de `TransferManager` pour télécharger un seul fichier (Amazon S3 objet) ou un répertoire (un Amazon S3 nom de compartiment suivi d'un préfixe d'objet) depuis Amazon S3.

Rubriques

- [Téléchargement d'un seul fichier](#)
- [Téléchargement d'un répertoire](#)

Téléchargement d'un seul fichier

Utilisez le gestionnaire de transfert `download`, fournissant la méthode `Amazon S3 Nom` du compartiment contenant l'objet que vous souhaitez télécharger, le nom de clé (objet) et un objet `Fichier` qui représente le fichier à créer sur votre système local.

Importations

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.Download;
import com.amazonaws.services.s3.transfer.MultipleFileDownload;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
```

```
import java.io.File;
```

Code

```
File f = new File(file_path);
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Download xfer = xfer_mgr.download(bucket_name, key_name, f);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Voir [Attente de la fin d'un transfert](#) Informations supplémentaires sur l'utilisation de `waitForCompletion` pour effectuer un transfert avec succès avant d'appeler `TransferManager.shutdownNow` Méthode. En attendant que le transfert se termine, vous pouvez interroger ou écouter les mises à jour relatives à son état et à sa progression. Pour plus d'informations, consultez [Obtention de l'état et de la progression du transfert](#).

Veillez consulter l'[exemple complet](#) sur GitHub.

Téléchargement d'un répertoire

Pour télécharger un ensemble de fichiers qui partagent un key prefix (analogue à un répertoire sur un système de fichiers) depuis Amazon S3, utilisez le `TransferManager.downloadDirectory` Méthode. Cette méthode accepte le nom du compartiment Amazon S3 contenant les objets que vous voulez télécharger, le préfixe d'objet partagé par tous les objets et un objet [File](#) qui représente le répertoire de votre système local dans lequel télécharger les fichiers. Si le répertoire nommé n'existe pas encore, il est créé.

Importations

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.transfer.Download;
import com.amazonaws.services.s3.transfer.MultipleFileDownload;
import com.amazonaws.services.s3.transfer.TransferManager;
```



```
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;

import java.io.File;
```

Code

```
TransferManager xfer_mgr = TransferManagerBuilder.standard().build();

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

Voir [Attente de la fin d'un transfert](#) Informations supplémentaires sur l'utilisation de `waitForCompletion` pour effectuer un transfert avec succès avant d'appeler `TransferManager.shutdownNow` Méthode. En attendant que le transfert se termine, vous pouvez interroger ou écouter les mises à jour relatives à son état et à sa progression. Pour plus d'informations, consultez [Obtention de l'état et de la progression du transfert](#).

Veillez consulter l'[exemple complet](#) sur GitHub.

Copie d'objets

Pour copier un objet depuis un compartiment S3 vers un autre, utilisez `TransferManager.copy` Méthode.

Importations

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

Code

```
System.out.println("Copying s3 object: " + from_key);
System.out.println("    from bucket: " + from_bucket);
System.out.println("    to s3 object: " + to_key);
System.out.println("    in bucket: " + to_bucket);

TransferManager xfer_mgr = TransferManagerBuilder.standard().build();
try {
    Copy xfer = xfer_mgr.copy(from_bucket, from_key, to_bucket, to_key);
    // loop with Transfer.isDone()
    XferMgrProgress.showTransferProgress(xfer);
    // or block with Transfer.waitForCompletion()
    XferMgrProgress.waitForCompletion(xfer);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
xfer_mgr.shutdownNow();
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Attente de la fin d'un transfert

Si votre application (ou thread) peut être bloquée jusqu'à ce que le transfert soit terminé, vous pouvez utiliser la [transfert](#) l'interface `waitForCompletion` pour bloquer jusqu'à ce que le transfert soit terminé ou qu'une exception se produise.

```
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);
}
```

Vous obtenez la progression des transferts si vous interrogez les événements avant l'appel de `waitForCompletion`, implémentez un mécanisme d'interrogation sur un thread distinct ou recevez les mises à jour de la progression de manière asynchrone en utilisant un [ProgressListener](#).

Veillez consulter l'[exemple complet](#) sur GitHub.

Obtention de l'état et de la progression du transfert

Chacune des classes renvoyées par le `TransferManager.upload*`, `download*`, `etcopy` renvoie une instance d'une des classes suivantes, selon qu'il s'agit d'une opération sur un fichier unique ou sur plusieurs fichiers.

Classe	Renvoyée par
Copy	<code>copy</code>
Download	<code>download</code>
MultipleFileDownload	<code>downloadDirectory</code>
Charger	<code>upload</code>
MultipleFileUpload	<code>uploadFileList</code> , <code>uploadDirectory</code>

Toutes ces classes implémentent l'interface [Transfer](#). `Transfer` fournit des méthodes utiles pour obtenir la progression d'un transfert, suspendre ou reprendre le transfert, et obtenir l'état actuel ou final du transfert.

Rubriques

- [Interrogation de la progression en cours d'un transfert](#)
- [Obtention de la progression d'un transfert avec un `ProgressListener`](#)
- [Obtention de la progression des sous-transferts](#)

Interrogation de la progression en cours d'un transfert

Cette boucle imprime la progression d'un transfert, examine sa progression en cours lors de l'exécution et, une fois le transfert terminé, imprime son état final.

Importations

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

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

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

Code

```
// print the transfer's human-readable description
System.out.println(xfer.getDescription());
// print an empty progress bar...
printProgressBar(0.0);
// update the progress bar while the xfer is ongoing.
do {
    try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        return;
    }
    // Note: so_far and total aren't used, they're just for
    // documentation purposes.
    TransferProgress progress = xfer.getProgress();
    long so_far = progress.getBytesTransferred();
    long total = progress.getTotalBytesToTransfer();
    double pct = progress.getPercentTransferred();
    eraseProgressBar();
    printProgressBar(pct);
} while (xfer.isDone() == false);
// print the final state of the transfer.
TransferState xfer_state = xfer.getState();
System.out.println(": " + xfer_state);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Obtention de la progression d'un transfert avec un ProgressListener

Vous pouvez attacher un objet [ProgressListener](#) vers n'importe quel transfert à l'aide de [la transfert](#) l'interface `addProgressListener` Méthode.

Un [ProgressListener](#) nécessite une seule méthode, `progressChanged`, qui accepte un objet [ProgressEvent](#). Vous pouvez utiliser l'objet pour obtenir le nombre total d'octets de l'opération en

appelant sa méthode `getBytes`, ainsi que le nombre d'octets transférés jusqu'à présent en appelant `getBytesTransferred`.

Importations

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

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

Code

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Obtention de la progression des sous-transferts

La classe [MultipleFileUpload](#) peut renvoyer des informations sur ses sous-transferts en appelant sa méthode `getSubTransfers`. Elle renvoie un objet non modifiable [Collection](#) de [Charger](#) objets qui fournissent l'état et la progression de chaque sous-transfert.

Importations

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

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

Code

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

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

// wait a bit before the next update.
try {
    Thread.sleep(200);
} catch (InterruptedException e) {
```

```
        return;
    }
} while (multi_upload.isDone() == false);
// print the final state of the transfer.
TransferState xfer_state = multi_upload.getState();
System.out.println("\nMultipleFileUpload " + xfer_state);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Plus d'informations

- [Clés d'objet](#) dans le Amazon Simple Storage Service Guide de l'utilisateur

Configuration d'un Amazon S3 Bucket en tant que site web

Vous pouvez configurer un compartiment Amazon S3 pour qu'il se comporte comme un site web. Pour ce faire, vous devez définir sa configuration de site web.

Note

Ces exemples de code supposent que vous comprenez le contenu de [Utilisation de AWS SDK for Java](#) et ont configuré la valeur par défaut AWS informations d'identification utilisant les informations de [Configuration AWS Informations d'identification et région pour le développement](#).

Définition de la configuration de site web d'un compartiment

Pour définir un Amazon S3 configuration du site Web du compartiment, appelez `setWebsiteConfiguration` avec le nom du compartiment pour définir la configuration, et un [BucketWebsiteConfiguration](#) objet contenant la configuration du site Web du compartiment.

La définition d'un document d'index est obligatoire ; tous les autres paramètres sont facultatifs.

Importations

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

```
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketWebsiteConfiguration;
```

Code

```
String bucket_name, String index_doc, String error_doc) {
BucketWebsiteConfiguration website_config = null;

if (index_doc == null) {
    website_config = new BucketWebsiteConfiguration();
} else if (error_doc == null) {
    website_config = new BucketWebsiteConfiguration(index_doc);
} else {
    website_config = new BucketWebsiteConfiguration(index_doc, error_doc);
}

final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.setBucketWebsiteConfiguration(bucket_name, website_config);
} catch (AmazonServiceException e) {
    System.out.format(
        "Failed to set website configuration for bucket '%s'\n",
        bucket_name);
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Note

La définition d'une configuration de site web ne modifie pas les autorisations d'accès de votre compartiment. Pour que vos fichiers soient visibles sur le web, vous devez également définir une stratégie de compartiment qui autorise l'accès en lecture public aux fichiers du compartiment. Pour de plus amples informations, veuillez consulter [Gestion de l'accès à Amazon S3 compartiments utilisant des stratégies de compartiment](#).

Veillez consulter l'[exemple complet](#) sur GitHub.

Obtention de la configuration de site web d'un compartiment

Obtentions d'unAmazon S3configuration du site Web du compartiment, appelez `getWebsiteConfiguration` avec le nom du compartiment pour laquelle récupérer la configuration.

La configuration est renvoyée sous la forme d'un objet [BucketWebsiteConfiguration](#). S'il n'y a pas de configuration de site web pour le compartiment, la valeur `null` est renvoyée.

Importations

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

Code

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

Veuillez consulter l'[exemple complet](#) sur GitHub.

Suppression de la configuration de site web d'un compartiment

Pour supprimer une Amazon S3 configuration du site Web du compartiment, appelez `deleteWebsiteConfiguration` avec le nom du compartiment dont la configuration doit être supprimée.

Importations

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

Code

```
final AmazonS3 s3 =
    AmazonS3ClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
try {
    s3.deleteBucketWebsiteConfiguration(bucket_name);
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.out.println("Failed to delete website configuration!");
    System.exit(1);
}
```

Veillez consulter l'[exemple complet](#) sur GitHub.

En savoir plus

- [PUT Bucket website](#) dans le Amazon S3 API Reference
- [GET Bucket website](#) dans le Amazon S3 API Reference
- [DELETE Bucket website](#) dans le Amazon S3 API Reference

Utiliser Amazon S3 Chiffrement côté client

Le chiffrement de données à l'aide du client de chiffrement Amazon S3 est un moyen de fournir une couche de protection supplémentaire pour les informations sensibles que vous stockez dans Amazon S3. Les exemples de cette section montrent comment créer et configurer le client de chiffrement Amazon S3 pour votre application.

Si vous découvrez le chiffrement, consultez le [Principes de base du chiffrement](#) dans le [AWSKMS Developer Guide](#) pour une présentation de base des termes et des algorithmes de chiffrement. Pour plus d'informations sur la prise en charge de la cryptographie [AWS Kits SDK](#), consultez la section [AWS Prise en charge des kits SDK pour Amazon S3 Chiffrement côté client](#) dans le [Amazon Web Services Références générales](#).

Note

Ces exemples de code supposent que vous comprenez le contenu de [Utilisation de AWS SDK for Java](#) et ont configuré la valeur par défaut [AWS Informations d'identification](#) utilisant les informations de [Configuration AWS Références et région pour le développement](#).

Si vous utilisez la version 1.11.836 ou antérieure du [AWS SDK for Java](#), consultez la section [Amazon S3 Migration client de chiffrement](#) pour plus d'informations sur la migration de vos applications vers des versions ultérieures. Si vous ne pouvez pas migrer, voir [cet exemple complet](#) sur GitHub.

Sinon, si vous utilisez la version 1.11.837 ou ultérieure du [AWS SDK for Java](#), explorez les exemples de rubriques énumérés ci-dessous pour utiliser [Amazon S3 Chiffrement côté client](#).

Rubriques

- [Amazon S3 Chiffrement côté client avec les clés principales du client](#)
- [Amazon S3 Chiffrement côté client avec AWS Clés gérées KMS](#)

Amazon S3 Chiffrement côté client avec les clés principales du client

Les exemples suivants s'appuyant sur le [Créateur de chiffrement Amazon S3 Client V2](#) pour créer une classe [Amazon S3 client](#) avec le chiffrement côté client activé. Une fois activés, les objets que vous chargez dans Amazon S3 à l'aide de ce client sont chiffrés. Les objets que vous obtenez d'Amazon S3 à l'aide de ce client sont automatiquement déchiffrés.

Note

Les exemples suivants illustrent l'utilisation du chiffrement côté client Amazon S3 avec des clés principales client gérées par les clients. Pour savoir comment utiliser le chiffrement avec [AWS Clés gérées KMS](#), voir [Amazon S3 Chiffrement côté client avec AWS Clés gérées KMS](#).

Au moment d'activer côté client, vous avez le choix entre deux modes de chiffrement Amazon S3 : strictement authentifié ou authentifié. Les sections suivantes montrent comment activer chaque type. Pour savoir quels sont les algorithmes utilisés par chaque mode, consultez la définition [CryptoMode](#).

Importations nécessaires

Pour ces exemples, vous devez importer les classes suivantes.

Importations

```
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;
```

chiffrement authentifié strict

Le chiffrement authentifié strict est le mode par défaut si aucun `CryptoMode` est spécifié.

Pour activer explicitement ce mode, spécifiez le `StrictAuthenticatedEncryption` valeur dans le `withCryptoConfiguration` méthode.

Note

Pour utiliser le chiffrement authentifié côté client, vous devez inclure le dernier fichier [jar Bouncy Castle](#) dans le chemin de classe de votre application.

Code

```
AmazonS3EncryptionV2 s3Encryption = AmazonS3EncryptionClientV2Builder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCryptoConfiguration(new
        CryptoConfigurationV2().withCryptoMode((CryptoMode.StrictAuthenticatedEncryption)))
    .withEncryptionMaterialsProvider(new StaticEncryptionMaterialsProvider(new
        EncryptionMaterials(secretKey)))
```

```
        .build();

s3Encryption.putObject(bucket_name, ENCRYPTED_KEY2, "This is the 2nd content to
encrypt");
```

Mode de chiffrement authentifié

Lorsque vous utilisez le mode `AuthenticatedEncryption`, un algorithme d'encapsulation de clé amélioré est appliqué pendant le chiffrement. Lorsque vous déchiffrez dans ce mode, l'algorithme peut vérifier l'intégrité de l'objet déchiffré et générer une exception si la vérification échoue. Pour de plus amples informations sur le fonctionnement du chiffrement authentifié, veuillez consulter le [Amazon S3Chiffrement authentifié côté client](#) billet de blog.

Note

Pour utiliser le chiffrement authentifié côté client, vous devez inclure le dernier fichier [jar Bouncy Castle](#) dans le chemin de classe de votre application.

Pour activer ce mode, spécifiez la valeur `AuthenticatedEncryption` dans la méthode `withCryptoConfiguration`.

Code

```
AmazonS3EncryptionV2 s3EncryptionClientV2 =
    AmazonS3EncryptionClientV2Builder.standard()
        .withRegion(Regions.DEFAULT_REGION)
        .withClientConfiguration(new ClientConfiguration())
        .withCryptoConfiguration(new
CryptoConfigurationV2().withCryptoMode(CryptoMode.AuthenticatedEncryption))
        .withEncryptionMaterialsProvider(new StaticEncryptionMaterialsProvider(new
EncryptionMaterials(secretKey)))
        .build();

s3EncryptionClientV2.putObject(bucket_name, ENCRYPTED_KEY1, "This is the 1st content to
encrypt");
```

Amazon S3Chiffrement côté client avec AWS Clés gérées KMS

Les exemples suivants utilisent le [Créateur de chiffrement Amazon S3 Client V2](#) pour créer un Amazon S3 client avec le chiffrement côté client activé. Une fois configurés, les objets que vous chargez dans

Amazon S3 à l'aide de ce client sont chiffrés. Les objets que vous obtenez d'Amazon S3 à partir de ce client sont automatiquement déchiffrés.

Note

Les exemples suivants montrent comment utiliser le `AmazonS3Chiffrement` côté client avec AWS Clés gérées par KMS. Pour en savoir plus sur l'utilisation du chiffrement avec vos propres clés, consultez [Amazon S3Chiffrement côté client avec des clés principales client](#).

Au moment d'activer côté client, vous avez le choix entre deux modes de chiffrement `AmazonS3Chiffrement` : strictement authentifié ou authentifié. Les sections suivantes montrent comment activer chaque type. Pour savoir quels sont les algorithmes utilisés par chaque mode, consultez la définition [CryptoMode](#).

Importations nécessaires

Pour ces exemples, vous devez importer les classes suivantes.

Importations

```
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;
```

chiffrement authentifié strict

Le chiffrement authentifié strict est le mode par défaut si aucun `CryptoMode` est spécifié.

Pour activer explicitement ce mode, spécifiez le `StrictAuthenticatedEncryption` valeur dans le `withCryptoConfiguration` Méthode.

Note

Pour utiliser le chiffrement authentifié côté client, vous devez inclure le dernier fichier [jar Bouncy Castle](#) dans le chemin de classe de votre application.

Code

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

s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
with a key created in the {console}");
System.out.println(s3Encryption.getObjectAsString(bucket_name, ENCRYPTED_KEY3));
```

Appelez la méthode `putObject` sur le client de chiffrement Amazon S3 pour charger des objets.

Code

```
s3Encryption.putObject(bucket_name, ENCRYPTED_KEY3, "This is the 3rd content to encrypt
with a key created in the {console}");
```

Vous pouvez récupérer l'objet en utilisant le même client. Cet exemple appelle la méthode `getObjectAsString` pour récupérer la chaîne qui a été stockée.

Code

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

Mode de chiffrement authentifié

Lorsque vous utilisez le mode `AuthenticatedEncryption`, un algorithme d'encapsulation de clé amélioré est appliqué pendant le chiffrement. Lorsque vous déchiffrez dans ce mode, l'algorithme peut vérifier l'intégrité de l'objet déchiffré et générer une exception si la vérification échoue. Pour plus d'informations sur le fonctionnement du chiffrement authentifié, consultez le [Amazon S3Chiffrement authentifié côté client](#) billet de blog.

Note

Pour utiliser le chiffrement authentifié côté client, vous devez inclure le dernier fichier [jar Bouncy Castle](#) dans le chemin de classe de votre application.

Pour activer ce mode, spécifiez la valeur `AuthenticatedEncryption` dans la méthode `withCryptoConfiguration`.

Code

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

Configuration deAWS KMSclient

LeAmazon S3client de chiffrement crée unAWS KMSclient par défaut, à moins qu'il ne soit explicitement spécifié.

Pour définir la région de cette zone créée automatiquementAWS KMSclient, définissez `leawsKmsRegion`.

Code

```
Region kmsRegion = Region.getRegion(Regions.AP_NORTHEAST_1);

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

Vous pouvez également utiliser le vôtreAWS KMSclient pour initialiser le client de chiffrement.

Code

```
AWSKMS kmsClient = AWKMSClientBuilder.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 Exemples avec utilisation de l’AWS SDK for Java

Cette section fournit des exemples de programmation d'[Amazon SQS](#) à l'aide du kit [AWS SDK for Java](#).

Note

Les exemples incluent uniquement le code nécessaire pour démontrer chaque technique. L'exemple de code complet [est disponible sur GitHub](#). À partir de là, vous pouvez télécharger un fichier source unique ou cloner le référentiel en local pour obtenir tous les exemples à générer et exécuter.

Rubriques

- [Utilisation d'Amazon SQS Feues d'attente de messages](#)
- [Envoi, réception et suppression Amazon SQS Messages](#)
- [Activation de l'interrogation longue pour Amazon SQS Files d'attente de messages](#)
- [Définition du délai de visibilité dans Amazon SQS](#)
- [Utilisation des files d'attente de lettres mortes dans Amazon SQS](#)

Utilisation d'Amazon SQS Feues d'attente de messages

Une file d'attente de messages est le conteneur logique utilisé pour envoyer en toute fiabilité des messages dans Amazon SQS. Il existe deux types de files d'attente : standard et FIFO (premier entré, premier sorti). Pour en savoir plus sur les files d'attente et les différences entre ces types, consultez le [Amazon SQS Manuel du développeur](#).

Cette rubrique explique comment créer, répertorier, supprimer une file d'attente Amazon SQS et obtenir son URL à l'aide du kit AWS SDK for Java.

Création d'une file d'attente

Utilisation du client `AmazonSQSCreateQueue`, fournissant un [CreateQueueRequest](#) qui décrit les paramètres de file d'attente.

Importations

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
```

Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
CreateQueueRequest create_request = new CreateQueueRequest(QueueName)
    .addAttributesEntry("DelaySeconds", "60")
    .addAttributesEntry("MessageRetentionPeriod", "86400");

try {
    sqs.createQueue(create_request);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Vous pouvez utiliser la forme simplifiée de `createQueue`, qui nécessite uniquement un nom de file d'attente, pour créer une file d'attente standard.

```
sqs.createQueue("MyQueue" + new Date().getTime());
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Affichage de la liste des files d'attente

Pour répertorier les files d'attente de votre compte, appelez le client `AmazonSQSListQueues` méthode.

Importations

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

Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
ListQueuesResult lq_result = sqs.listQueues();  
System.out.println("Your SQS Queue URLs:");  
for (String url : lq_result.getQueueUrls()) {  
    System.out.println(url);  
}
```

L'utilisation de la surcharge `listQueues` sans aucun paramètre renvoie toutes les files d'attente. Vous pouvez filtrer les résultats renvoyés en transmettant un objet `ListQueuesRequest`.

Importations

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

Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
String name_prefix = "Queue";  
lq_result = sqs.listQueues(new ListQueuesRequest(name_prefix));  
System.out.println("Queue URLs with prefix: " + name_prefix);  
for (String url : lq_result.getQueueUrls()) {  
    System.out.println(url);  
}
```

Veuillez consulter l'[exemple complet](#) sur GitHub.

Obtention de l'URL d'une file d'attente

Appellez le client `AmazonSQS` `getQueueUrl` Méthode.

Importations

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

Code

```
AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();
String queue_url = sqs.getQueueUrl(QueueName).getQueueUrl();
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Suppression d'une file d'attente

Fournissez la file d'attente [URL](#) vers le client `AmazonSQS.deleteQueue` Méthode.

Importations

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

Code

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Plus d'informations

- [Comment Amazon SQS Feux d'attente](#) dans le Amazon SQS Manuel du développeur
- [CreateQueue](#) dans le Amazon SQS API Reference
- [GetQueueUrl](#) dans le Amazon SQS API Reference
- [ListQueues](#) dans le Amazon SQS API Reference
- [DeleteQueues](#) dans le Amazon SQS API Reference

Envoi, réception et suppression Amazon SQS Messages

Cette rubrique explique comment envoyer, recevoir et supprimer des messages Amazon SQS. Les messages sont toujours livrés à l'aide d'une [file d'attente SQS](#).

Envoi d'un message

Ajoutez un simple message à un Amazon SQS file d'attente en appelant le client `AmazonSQS.sendMessage` méthode. Fournissez un objet [SendMessageRequest](#) qui contient l'[URL](#) de la file d'attente, le corps du message et une valeur de délai d'annulation (en secondes).

Importations

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

Code

```
SendMessageRequest send_msg_request = new SendMessageRequest()
    .withQueueUrl(queueUrl)
    .withMessageBody("hello world")
    .withDelaySeconds(5);
sqs.sendMessage(send_msg_request);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Envoi simultané de plusieurs messages

Vous pouvez envoyer plusieurs message dans une même demande. Pour envoyer plusieurs messages, utilisez `sendMessageBatch`, qui prend une [SendMessageBatchRequest](#) contenant l'URL de file d'attente et la liste des messages (chacun d'un [SendMessageBatchRequestEntry](#)) à envoyer. Vous pouvez également définir une valeur de délai facultative par message.

Importations

```
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
import com.amazonaws.services.sqs.model.SendMessageBatchRequestEntry;
```

Code

```
SendMessageBatchRequest send_batch_request = new SendMessageBatchRequest()
    .withQueueUrl(queueUrl)
    .withEntries(
        new SendMessageBatchRequestEntry(
            "msg_1", "Hello from message 1"),
```

```
new SendMessageBatchRequestEntry(
    "msg_2", "Hello from message 2")
    .withDelaySeconds(10));
sqs.sendMessageBatch(send_batch_request);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Réception de messages

Récupérez tous les messages qui sont actuellement dans la file d'attente en appelant `receiveMessage`, lui transmettant l'URL de la file d'attente. Les messages sont renvoyés sous la forme d'une liste d'objets [Message](#).

Importations

```
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.SendMessageBatchRequest;
```

Code

```
List<Message> messages = sqs.receiveMessage(queueUrl).getMessages();
```

Suppression des messages après réception

Après la réception d'un message et le traitement de son contenu, supprimez le message de la file d'attente en envoyant le descripteur de réception du message et l'URL de file d'attente au client `AmazonSQS.deleteMessage` méthode.

Code

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Plus d'informations

- [Comment Amazon SQS Travail des files d'attente](#) dans le Amazon SQS Manuel du développeur
- [SendMessage](#) dans le Amazon SQS API Reference

- [SendMessageBatch](#) dans le Amazon SQS API Reference
- [ReceiveMessage](#) dans le Amazon SQS API Reference
- [DeleteMessage](#) dans le Amazon SQS API Reference

Activation de l'interrogation longue pour Amazon SQS Files d'attente de messages

Amazon SQS les usages interrogation courte par défaut, en interrogeant uniquement un sous-ensemble des serveurs (selon une distribution aléatoire pondérée) pour déterminer si des messages peuvent être inclus dans la réponse.

Les sondages longs permettent de réduire les coûts d'utilisation Amazon SQS en réduisant le nombre de réponses vides lorsqu'il n'y a pas de messages à renvoyer en réponse à un `ReceiveMessage` demande envoyée à un Amazon SQS file d'attente et élimination des réponses fausses vides.

Note

Vous pouvez définir une fréquence d'attente active de longue durée à partir de 1 à 20 secondes.

Activation de l'attente active de longue durée lors de la création d'une file d'attente

Pour activer l'interrogation longue lors de la création d'un Amazon SQS file d'attente, définissez la `ReceiveMessageWaitTimeSeconds` attribut sur l'objet [CreateQueueRequest](#) objet avant d'appeler la classe `AmazonSQS createQueue` méthode.

Imports

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.AmazonSQSException;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
```

Code

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

// Enable long polling when creating a queue
```

```
CreateQueueRequest create_request = new CreateQueueRequest()
    .withQueueName(queue_name)
    .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");

try {
    sqs.createQueue(create_request);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Activation de l'attente active de longue durée pour une file d'attente existante

En plus de l'activation de l'attente active de longue durée lors de la création d'une file d'attente, vous pouvez également l'activer sur une file d'attente existante en définissant `ReceiveMessageWaitTimeSeconds` sur le [SetQueueAttributesRequest](#) avant d'appeler la classe `AmazonSQSsetQueueAttributes` Méthode.

Importations

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

Code

```
SetQueueAttributesRequest set_attrs_request = new SetQueueAttributesRequest()
    .withQueueUrl(queue_url)
    .addAttributesEntry("ReceiveMessageWaitTimeSeconds", "20");
sqs.setQueueAttributes(set_attrs_request);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Activation de l'attente active de longue durée pour la réception des messages

Vous pouvez activer l'interrogation longue lors de la réception d'un message en définissant le temps d'attente en secondes sur le [ReceiveMessageRequest](#) que vous fournissez à la classe `AmazonSQSreceiveMessage` Méthode.

Note

Vous devez vous assurer que le délai d'expiration de la demande du client est supérieur à la durée maximale d'attente active de longue durée (20 secondes) de manière à ce que vos requêtes ne demandent pas à attendre le prochain événement de sondage !

Importations

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

Code

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

Veillez consulter l'[exemple complet](#) sur GitHub.

Plus d'informations

- [Amazon SQS Attente active de longue durée d'](#) dans le Amazon SQS Manuel du développeur
- [CreateQueue](#) dans le Amazon SQS API Reference
- [ReceiveMessage](#) dans le Amazon SQS API Reference
- [SetQueueAttributes](#) dans le Amazon SQS API Reference

Définition du délai de visibilité dans Amazon SQS

Lorsqu'un message est reçu dans Amazon SQS, il demeure dans la file d'attente jusqu'à ce qu'il soit supprimé, afin de garantir la réception. Un message qui a été reçu, mais pas supprimé, est disponible dans les demandes suivantes après un délai de visibilité donné afin d'empêcher que le message ne soit reçu plusieurs fois avant d'être traité et supprimé.

Note

Lorsque vous utilisez les [files d'attente standard](#), le délai de visibilité n'est pas une garantie que vous ne recevrez pas deux fois un même message. Si vous utilisez une file d'attente standard, assurez-vous que votre code gère le cas où le même message est remis plusieurs fois.

Définition du délai de visibilité de message pour un seul message

Lorsque vous avez reçu un message, vous pouvez modifier son délai de visibilité en transmettant son descripteur de réception dans un objet [ChangeMessageVisibilityRequest](#) que vous passez à la classe AmazonSQS » `changeMessageVisibility` méthode.

Importations

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

Code

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

// Get the receipt handle for the first message in the queue.
String receipt = sqs.receiveMessage(queue_url)
    .getMessages()
    .get(0)
    .getReceiptHandle();

sqs.changeMessageVisibility(queue_url, receipt, timeout);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Définition simultanée du délai de visibilité de message pour plusieurs messages

Pour définir simultanément le délai de visibilité de plusieurs messages, créez une liste d'objets [ChangeMessageVisibilityBatchRequestEntry](#), contenant chacun une chaîne d'ID unique et un descripteur de réception. Transmettez ensuite la liste à la méthode `changeMessageVisibilityBatch` de la classe client Amazon SQS.

Importations

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.ChangeMessageVisibilityBatchRequestEntry;
import java.util.ArrayList;
import java.util.List;
```

Code

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

List<ChangeMessageVisibilityBatchRequestEntry> entries =
    new ArrayList<ChangeMessageVisibilityBatchRequestEntry>();

entries.add(new ChangeMessageVisibilityBatchRequestEntry(
    "unique_id_msg1",
    sqs.receiveMessage(queue_url)
        .getMessages()
        .get(0)
        .getReceiptHandle())
    .withVisibilityTimeout(timeout));

entries.add(new ChangeMessageVisibilityBatchRequestEntry(
    "unique_id_msg2",
    sqs.receiveMessage(queue_url)
        .getMessages()
        .get(0)
        .getReceiptHandle())
    .withVisibilityTimeout(timeout + 200));

sqs.changeMessageVisibilityBatch(queue_url, entries);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Plus d'informations

- [Délai de visibilité](#) dans le Amazon SQS Manuel du développeur
- [SetQueueAttributes](#) dans le Amazon SQS API Reference
- [GetQueueAttributes](#) dans le Amazon SQS API Reference
- [ReceiveMessage](#) dans le Amazon SQS API Reference

- [ChangeMessageVisibility](#) dans le Amazon SQS API Reference
- [ChangeMessageVisibilityBatch](#) dans le Amazon SQS API Reference

Utilisation des files d'attente de lettres mortes dans Amazon SQS

Amazon SQS prend en charge les files d'attente de lettres mortes. Il s'agit d'une file d'attente que peuvent cibler d'autres files d'attente (source) pour les messages qui ne sont pas traités avec succès. Vous pouvez mettre de côté et isoler ces messages dans la file d'attente de lettres mortes pour déterminer pourquoi leur traitement a échoué.

Création d'une file d'attente de lettres mortes

Une file d'attente de lettres mortes est créée de la même manière qu'une file d'attente normale, mais elle comporte les restrictions suivantes :

- Une file d'attente de lettres mortes doit avoir le même type de file d'attente (FIFO ou standard) que la file d'attente source.
- Une file d'attente de lettres mortes doit être créée à l'aide de cette dernière Compte AWS Set région comme file d'attente source.

Ici, nous allons créer deux files d'attente Amazon SQS identiques, l'une d'entre elles tenant lieu de file d'attente de lettres mortes :

Importations

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

Code

```
final AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();  
  
// Create source queue  
try {  
    sqs.createQueue(src_queue_name);  
} catch (AmazonSQSException e) {  
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
```

```
        throw e;
    }
}

// Create dead-letter queue
try {
    sqs.createQueue(dl_queue_name);
} catch (AmazonSQSException e) {
    if (!e.getErrorCode().equals("QueueAlreadyExists")) {
        throw e;
    }
}
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Désignation d'une file d'attente de lettres mortes pour une file d'attente source

Pour désigner une file d'attente de lettres mortes, vous devez commencer par créer une stratégie de redirection, puis définir la stratégie dans les attributs de la file d'attente. Une stratégie de redirection est spécifiée au format JSON. Elle indique l'ARN de la file d'attente de lettres mortes et le nombre maximum de fois où le message peut être reçu et non traité avant d'être envoyé dans la file d'attente de lettres mortes.

Pour définir la stratégie de reconduite de votre file d'attente source, appelez la classe `AmazonSQS` »`setQueueAttributes` Méthode avec un [SetQueueAttributesRequest](#) objet pour lequel vous avez défini le `RedrivePolicy` avec votre stratégie de redrive JSON.

Importations

```
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
```

Code

```
String dl_queue_url = sqs.getQueueUrl(dl_queue_name)
    .getQueueUrl();

GetQueueAttributesResult queue_attrs = sqs.getQueueAttributes(
    new GetQueueAttributesRequest(dl_queue_url)
    .withAttributeNames("QueueArn"));
```

```
String dl_queue_arn = queue_attrs.getAttributes().get("QueueArn");

// Set dead letter queue with redrive policy on source queue.
String src_queue_url = sqs.getQueueUrl(src_queue_name)
    .getQueueUrl();

SetQueueAttributesRequest request = new SetQueueAttributesRequest()
    .withQueueUrl(src_queue_url)
    .addAttributesEntry("RedrivePolicy",
        "{\"maxReceiveCount\": \"5\", \"deadLetterTargetArn\": \""
        + dl_queue_arn + "\"}");

sqs.setQueueAttributes(request);
```

Veillez consulter l'[exemple complet](#) sur GitHub.

Plus d'informations

- [A l'aide de Amazon SQS Files d'attente de lettres mortes](#) dans le Amazon SQS Manuel du développeur
- [SetQueueAttributes](#) dans le Amazon SQS API Reference

Amazon SWF Exemples avec utilisation de l'AWS SDK for Java

[Amazon SWF](#) est un service de gestion des flux de travail qui permet aux développeurs de générer et de dimensionner des flux de travail distribués qui peuvent avoir des étapes parallèles ou séquentielles composées d'activités, de flux de travail enfants ou même de flux de travail enfants. [Lambda](#) tâches.

Il existe deux façons d'utiliser avec Amazon SWF Utilisation de AWS SDK for Java, en utilisant le SWF client ou en utilisant le AWS Flow Framework pour Java. Le AWS Flow Framework pour Java est plus difficile à configurer au départ, car il utilise massivement les annotations et s'appuie sur des bibliothèques supplémentaires telles qu'AspectJ et Spring Framework. Cependant, pour les projets volumineux ou complexes, vous gagnerez en temps de codage si vous utilisez AWS Flow Framework pour Java. Pour plus d'informations, consultez le [AWS Flow Framework Manuel du développeur pour Java](#).

Cette section propose des exemples de programmation d'Amazon SWF en utilisant directement le client AWS SDK for Java.

Rubriques

- [Principes de base SWF](#)
- [Création d'une Amazon SWF application simple](#)
- [Tâches Lambda](#)
- [Arrêt normal des travaux d'activité et de flux de travail](#)
- [Enregistrement de domaines](#)
- [Affichage des domaines](#)

Principes de base SWF

Il s'agit des modèles généraux d'utilisation d'Amazon SWF avec le kit AWS SDK for Java. Ils sont principalement destinés à servir de référence. Pour un didacticiel de présentation plus complet, consultez [Création d'un jeu simple Amazon SWF Application](#).

Dépendances

Les applications Amazon SWF de base nécessitent les dépendances suivantes, qui sont fournies avec le kit 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

les numéros de versions de ces packages dépendent de la version du kit SDK en votre disposition, mais les versions fournies avec le kit SDK ont été testées en termes de compatibilité et sont celles que vous devez utiliser.

AWS Flow Framework pour les applications Java nécessitent une configuration supplémentaire, et dépendances supplémentaires. Consultez [AWS Flow Framework Manuel du développeur de Java](#) pour plus d'informations sur l'utilisation du framework.

Importations

En général, vous pouvez utiliser les importations suivantes pour le développement du code :

```
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Une bonne pratique consiste néanmoins à importer uniquement les classes dont vous avez besoin. Vous vous retrouverez probablement à spécifier des classes particulières dans l'espace de travail `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;
```

Si vous utilisez le kit AWS Flow Framework pour Java, vous importez les classes depuis le `com.amazonaws.services.simpleworkflow.flow` espace de travail. Par Exemple:

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

Note

Le AWS Flow Framework présente d'autres conditions requises en plus de celles du kit de base AWS SDK for Java. Pour plus d'informations, consultez le [AWS Flow Framework Manuel du développeur de Java](#).

Utilisation de la classe client SWF

Votre interface de base avec Amazon SWF s'effectue via la classe [AmazonSimpleWorkflowClient](#) ou [AmazonSimpleWorkflowAsyncClient](#). La principale différence entre les deux classes est que la classe `*AsyncClient` renvoie des objets [Future](#) pour la programmation simultanée (asynchrone).


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

Création d'une Amazon SWF application simple

Cette rubrique vous présentera la programmation d'[Amazon SWF](#) applications avec le AWS SDK for Java, tout en présentant quelques concepts importants en cours de route.

À propos de l'exemple

L'exemple de projet créera un flux de travail avec une seule activité qui accepte les données du flux de travail transmises via le AWS cloud (selon la tradition `HelloWorld`, ce sera le nom de la personne à saluer), puis imprime un message d'accueil en réponse.

Si cela semble très simple au premier abord, les applications Amazon SWF sont constituées de différentes parties qui fonctionnent ensemble :

- Un domaine, utilisé comme conteneur logique pour vos données d'exécution de flux de travail.
- Un ou plusieurs flux de travail représentant des composants de code qui définissent l'ordre logique de l'exécution des activités du flux de travail et des flux de travail enfants.
- Un travail de flux de travail, également appelé décideur, qui recherche les tâches de décision et planifie des activités ou des flux de travail enfants en réponse.
- Une ou plusieurs activités, chacune représentant une unité de travail dans le flux de travail.
- Un travail d'activité qui recherche les tâches d'activité et exécute des méthodes d'activité en réponse.
- Une ou plusieurs listes de tâches, qui sont des files d'attente gérées par Amazon SWF servant à envoyer des demandes aux travaux de flux de travail et d'activité. Les tâches d'une liste de tâches destinées aux travaux de flux de travail sont appelées tâches de décision. Celles destinées aux travaux d'activité sont appelées tâches d'activité.
- Un démarreur de flux de travail qui démarre l'exécution de votre flux de travail.

Dans les coulisses, Amazon SWF orchestre le fonctionnement de ces composants, coordonne leur flux depuis le AWS cloud, transmet des données entre eux, gère les délais d'attente et les notifications de pulsations, et enregistre l'historique d'exécution du flux de travail.

Prérequis

Environnement de développement

L'environnement de développement utilisé dans ce didacticiel comprend les éléments suivants :

- La valeur [AWS SDK for Java](#).
- [Apache Maven](#) (3.3.1).
- JDK 1.7 ou version ultérieure. Ce didacticiel a été développé et testé à l'aide de JDK 1.8.0.
- Un éditeur de texte Java efficace (de votre choix).

Note

Si vous utilisez un système de génération différent de celui de Maven, vous pouvez toujours créer un projet en suivant les étapes appropriées à votre environnement et en utilisant les concepts fournis ici pour poursuivre. Vous trouverez plus d'informations sur la configuration et l'utilisation du kit AWS SDK for Java avec différents systèmes de génération dans [Mise en route](#).

De même, mais avec plus d'efforts, les étapes présentées ici peuvent être mises en œuvre à l'aide de n'importe quel AWS SDK compatible avec Amazon SWF

Toutes les dépendances externes nécessaires sont incluses avec le kit AWS SDK for Java. Vous n'avez donc aucun élément supplémentaire à télécharger.

AWSAccès

Pour suivre correctement ce didacticiel, vous devez avoir accès au portail AWS d'accès comme [décrit dans la section de configuration de base](#) de ce guide.

Les instructions décrivent comment accéder aux informations d'identification temporaires que vous copiez et collez dans votre `credentials` fichier partagé local. Les informations d'identification temporaires que vous collez doivent être associées à un rôle IAM autorisé à accéder à Amazon SWF. AWS IAM Identity Center Après avoir collé les informations d'identification temporaires, votre `credentials` fichier ressemblera à ce qui suit.

```
[default]
```


3. Assurez-vous que Maven génère votre projet avec JDK 1.7 ou version ultérieure. Ajoutez les éléments suivants à votre projet (avant ou après le bloc `<dependencies>`) dans `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>
```

Codage du projet

L'exemple de projet est composé de quatre applications distinctes que nous allons examiner une par une :

- `HelloTypes.java` --contient les données de domaine, d'activité et de type de flux de travail du projet, partagées avec les autres composants. Gère également l'enregistrement de ces types auprès de SWF.
- `ActivityWorker.java` --contient le moteur d'activité, qui interroge les tâches d'activité et exécute les activités en réponse.
- `WorkflowWorker.java` --contient le workflow worker (décideur), qui interroge les tâches de décision et planifie de nouvelles activités.
- `WorkflowStarter.java` --contient le démarreur du flux de travail, qui lance l'exécution d'un nouveau flux de travail, ce qui amène SWF à générer des tâches de décision et de flux de travail destinées à vos collaborateurs.

Étapes communes pour tous les fichiers source

Tous les fichiers que vous créez pour héberger vos classes Java présentent quelques points communs. Par souci de concision, ces étapes sont implicites chaque fois que vous ajoutez un nouveau fichier au projet :

1. Créez le fichier dans le répertoire `src/main/java/aws/example/helloswf/` du projet.
2. Ajoutez une déclaration `package` au début de chaque fichier pour déclarer son espace de noms. L'exemple de projet utilise :

```
package aws.example.helloswf;
```

3. Ajoutez `import` des déclarations pour la [AmazonSimpleWorkflowClient](#) classe et pour plusieurs classes dans l'espace de `com.amazonaws.services.simpleworkflow.model` nommage. Pour simplifier, nous utilisons :

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

Enregistrement d'un domaine, et de types de flux de travail et d'activité

Nous allons commencer par créer une nouvelle classe exécutable, `HelloTypes.java`. Ce fichier contient des données partagées que les différentes parties de votre flux de travail doivent connaître, comme le nom et la version des types d'activité et de flux de travail, le nom du domaine et le nom de la liste de tâches.

1. Ouvrez votre éditeur de texte et créez le fichier `HelloTypes.java`, en ajoutant une déclaration de `package` et des déclarations d'importation conformément aux [étapes courantes](#).
2. Déclarez la classe `HelloTypes` et fournissez-lui des valeurs à utiliser pour vos types d'activité et de flux de travail enregistrés :

```
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";
```

Ces valeurs seront utilisées dans l'ensemble du code.

3. Après les déclarations `String`, créez une instance de la [AmazonSimpleWorkflowClient](#) classe. Il s'agit de l'interface de base pour les méthodes Amazon SWF fournies par le kit AWS SDK for Java.

```
private static final AmazonSimpleWorkflow swf =  
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

L'extrait de code précédent suppose que des informations d'identification temporaires sont associées au default profil. Si vous utilisez un profil différent, modifiez le code ci-dessus comme suit et remplacez *profile_name* par le nom du profil actuel.

```
private static final AmazonSimpleWorkflow swf =  
    AmazonSimpleWorkflowClientBuilder  
        .standard()  
        .withCredentials(new ProfileCredentialsProvider("profile_name"))  
        .withRegion(Regions.DEFAULT_REGION)  
        .build();
```

4. Ajoutez une nouvelle fonction pour enregistrer un domaine SWF. Un domaine est un conteneur logique pour différents types d'activité et de flux de travail SWF associés. Les composants SWF ne peuvent communiquer entre eux que s'ils sont situés dans le même domaine.

```
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!");  
}
```

Lorsque vous enregistrez un domaine, vous lui fournissez un nom (un ensemble de 1 à 256 caractères : /|, à l'exclusion des caractères de contrôle ou de la chaîne littérale « arn ») et une période de conservation, qui correspond au nombre de jours pendant lesquels les données d'historique d'exécution de votre flux de travail Amazon SWF seront conservées une fois l'exécution d'un flux de travail terminée. La période de conservation maximale pour les exécutions de flux de travail est de 90 jours. [RegisterDomainRequest](#) Pour plus d'informations, voir.

Si un domaine portant ce nom existe déjà, un [DomainAlreadyExistsException](#) est généré. Comme cela ne pose aucun problème si le domaine a déjà été créé, nous pouvons ignorer cette exception.

Note

Ce code illustre un modèle courant lorsque des méthodes AWS SDK for Java sont utilisées. Les données de la méthode sont fournies par une classe dans l'espace de noms `simpleworkflow.model`, que vous instanciez et remplissez à l'aide des méthodes `@with*` chaînables.

5. Ajoutez une fonction pour enregistrer un nouveau type d'activité. Une activité représente une unité de travail de votre flux de travail.

```
try {
    System.out.println("** Registering the activity type '" + ACTIVITY +
        "-" + ACTIVITY_VERSION + "'.");
    swf.registerActivityType(new RegisterActivityTypeRequest()
        .withDomain(DOMAIN)
        .withName(ACTIVITY)
        .withVersion(ACTIVITY_VERSION)
        .withDefaultTaskList(new TaskList().withName(TASKLIST))
        .withDefaultTaskScheduleToStartTimeout("30")
        .withDefaultTaskStartToCloseTimeout("600")
        .withDefaultTaskScheduleToCloseTimeout("630")
        .withDefaultTaskHeartbeatTimeout("10"));
} catch (TypeAlreadyExistsException e) {
    System.out.println("** Activity type already exists!");
}
```

Un type d'activité est identifié par un nom et une version, qui sont utilisés pour identifier l'activité de façon unique parmi toutes les autres activités du domaine dans lequel cette activité est enregistrée. Les activités contiennent également différents paramètres facultatifs, comme la liste de tâches par défaut utilisée pour recevoir des tâches et des données provenant de SWF, ainsi que différents délais d'expiration permettant d'appliquer des contraintes quant à la durée de l'exécution de différentes parties de l'activité. [RegisterActivityTypeRequest](#) Pour plus d'informations, voir.

Note

Toutes les valeurs de délai sont spécifiées en secondes. Consultez la section [Types de Amazon SWF délais](#) d'expiration pour une description complète de la façon dont les délais d'expiration affectent l'exécution de votre flux de travail.

Si le type d'activité que vous essayez d'enregistrer existe déjà, un [TypeAlreadyExistsException](#) est déclenché. Ajoutez une fonction pour enregistrer un nouveau type de flux de travail. Un flux de travail, également appelé décideur, représente la logique de l'exécution de votre flux de travail.

+

```
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!");
}
```

+

À l'instar de types d'activité, les types de flux de travail sont identifiés par un nom et une version, et sont également associés à des délais d'expiration configurables. [RegisterWorkflowTypeRequest](#) Pour plus d'informations, voir.

+

Si le type de flux de travail que vous essayez d'enregistrer existe déjà, un [TypeAlreadyExistsException](#) est déclenché. Enfin, rendez la classe exécutable en lui fournissant une méthode `main` qui enregistre à son tour le domaine, le type d'activité et le type de flux de travail :

+


```
registerDomain();
registerWorkflowType();
registerActivityType();
```

Vous pouvez [créer](#) et [exécuter](#) l'application maintenant pour exécuter le script d'enregistrement, ou continuer à coder les travaux d'activité et de flux de travail. Une fois que le domaine, le flux de travail et l'activité ont été enregistrés, vous n'avez pas besoin de les exécuter à nouveau. Ces types persistent jusqu'à ce que vous les supprimiez vous-même.

Implémentation du travail d'activité

Une activité est l'unité de travail de base d'un flux de travail. Un flux de travail fournit la logique en planifiant les activités à exécuter (ou les autres actions à effectuer) en réponse à des tâches de décision. Un flux de travail classique comporte généralement différentes activités qui peuvent s'exécuter de façon synchrone ou asynchrone, ou avec une combinaison de ces deux modes.

Le travail d'activité est le fragment de code qui recherche les tâches d'activité qui sont générées par Amazon SWF en réponse à des décisions de flux de travail. Lorsque ce travail reçoit une tâche d'activité, il exécute l'activité correspondante et renvoie une réponse de réussite/échec au flux de travail.

Nous allons implémenter un travail d'activité simple qui traite une activité unique.

1. Ouvrez votre éditeur de texte et créez le fichier `ActivityWorker.java`, en ajoutant une déclaration de package et des déclarations d'importation conformément aux [étapes courantes](#).

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflow;
import com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClientBuilder;
import com.amazonaws.services.simpleworkflow.model.*;
```

2. Ajoutez la classe `ActivityWorker` au fichier et fournissez-lui un membre de données pour stocker un client SWF que nous utiliserons pour interagir avec Amazon SWF :

```
private static final AmazonSimpleWorkflow swf =
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

3. Ajoutez la méthode que nous utiliserons comme activité :

```
private static String sayHello(String input) throws Throwable {
```

```
    return "Hello, " + input + "!";  
}
```

L'activité prend simplement une chaîne, la combine en une salutation et renvoie le résultat. Même si les risques que cette activité lève une exception sont minimes, il est judicieux de concevoir des activités qui peuvent générer une erreur en cas de problème.

4. Ajoutez une méthode `main` que nous utiliserons comme méthode d'interrogation des tâches d'activité. Nous démarrons cette méthode en ajoutant du code pour rechercher des tâches d'activité dans la liste des tâches :

```
System.out.println("Polling for an activity task from the tasklist '"  
    + HelloTypes.TASKLIST + "' in the domain '"  
    + HelloTypes.DOMAIN + "'.");  
  
ActivityTask task = swf.pollForActivityTask(  
    new PollForActivityTaskRequest()  
        .withDomain(HelloTypes.DOMAIN)  
        .withTaskList(  
            new TaskList().withName(HelloTypes.TASKLIST)));  
  
String task_token = task.getTaskToken();
```

L'activité reçoit des tâches Amazon SWF en appelant la `pollForActivityTask` méthode du client SWF, en spécifiant le domaine et la liste de tâches à utiliser dans le fichier transmis [PollForActivityTaskRequest](#).

Une fois qu'une tâche est reçue, nous récupérons un identificateur unique pour celle-ci en appelant la méthode `getTaskToken` de la tâche.

5. Ensuite, écrivez du code pour traiter les tâches qui arrivent. Ajoutez ce qui suit à votre méthode `main` juste après le code qui recherche la tâche et récupère son jeton.

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

Si le jeton de la tâche n'est pas null, nous pouvons commencer à exécuter la méthode d'activité (`sayHello`) en lui fournissant les données d'entrée qui ont été envoyées avec la tâche.

Si la tâche aboutit (aucune erreur n'a été générée), le travailleur répond au SWF en appelant la `respondActivityTaskCompleted` méthode du client SWF avec un [RespondActivityTaskCompletedRequest](#) objet contenant le jeton de tâche et les données de résultat de l'activité.

En revanche, si la tâche échoue, nous répondons en appelant la `respondActivityTaskFailed` méthode avec un [RespondActivityTaskFailedRequest](#) objet, en lui transmettant le jeton de tâche et des informations sur l'erreur.

Note

Cette activité ne s'arrêtera pas correctement si elle est supprimée. Même si cela dépasse le cadre de ce didacticiel, une autre implémentation de ce travail d'activité est fournie dans la rubrique connexe [Arrêt normal des travaux d'activité et de flux de travail](#).

Implémentation du travail de flux de travail

La logique de flux de travail est située dans un élément de code appelé travail de flux de travail. Le travail de flux de travail recherche les tâches de décision envoyées par Amazon SWF dans le domaine et dans la liste de tâches par défaut avec lesquels le type de flux de travail a été enregistré.

Lorsque le travail de flux de travail reçoit une tâche, il prend une décision (il s'agit généralement de décider s'il faut ou non planifier une nouvelle activité) et exécute une action appropriée (par exemple, planifier l'activité).

1. Ouvrez votre éditeur de texte et créez le fichier `WorkflowWorker.java`, en ajoutant une déclaration de package et des déclarations d'importation conformément aux [étapes courantes](#).
2. Ajoutez quelques déclarations d'importation supplémentaires au fichier :

```
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. Déclarez la `WorkflowWorker` classe et créez une instance de la [AmazonSimpleWorkflowClient](#) classe utilisée pour accéder aux méthodes SWF.

```
private static final AmazonSimpleWorkflow swf =
    AmazonSimpleWorkflowClientBuilder.standard().withRegion(Regions.DEFAULT_REGION).build();
```

4. Ajoutez la méthode `main`. Cette méthode exécute une boucle continue en recherchant les tâches de décision à l'aide de la méthode `pollForDecisionTask` du client SWF. Le [PollForDecisionTaskRequest](#) fournit les détails.

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

Une fois qu'une tâche est reçue, nous appelons sa méthode `getTaskToken` qui renvoie une chaîne permettant d'identifier la tâche. Si le jeton renvoyé ne l'est pas `null`, nous le traitons plus avant dans la `executeDecisionTask` méthode, en lui transmettant le jeton de tâche et la liste des [HistoryEvent](#) objets envoyés avec la tâche.

5. Ajoutez la méthode `executeDecisionTask`, en prenant le jeton de la tâche (un élément `String`) et la liste `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;
```

Nous configurons également des membres de données pour suivre des éléments comme :

- Une liste d'objets [Decision](#) utilisés pour signaler les résultats du traitement de la tâche.
- Une chaîne contenant l'entrée du flux de travail fournie par l'événement `WorkflowExecutionStarted` « »
- Un comptage des activités planifiées et ouvertes (en cours d'exécution) pour éviter de planifier la même activité lorsque celle-ci a déjà été planifiée ou est en cours d'exécution.
- Une valeur booléenne pour indiquer que l'activité est terminée.
- Une chaîne pour stocker les résultats de l'activité, qui sera renvoyée en tant que résultat de notre flux de travail.

6. Ajoutez ensuite du code à `executeDecisionTask` pour traiter les objets `HistoryEvent` qui ont été envoyés avec la tâche, en fonction du type d'événement signalé par la méthode `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("]");
```

Dans le cadre de notre flux de travail, voici les éléments qui présentent le plus d'intérêt pour nous :

- l'événement « `WorkflowExecutionStarted` », qui indique que l'exécution du flux de travail a commencé (ce qui signifie généralement que vous devez exécuter la première activité du flux

de travail) et qui fournit l'entrée initiale fournie au flux de travail. Dans ce cas, il s'agit de la partie nom de notre salutation. Cet événement est donc enregistré dans une chaîne à utiliser lors de la planification de l'activité à exécuter.

- l'événement « `ActivityTaskCompleted` », qui est envoyé une fois que l'activité planifiée est terminée. Les données d'événement comprennent également la valeur de retour de l'activité terminée. Comme nous n'avons qu'une seule activité, nous utiliserons cette valeur comme résultat de la totalité du flux de travail.

Les autres types d'événement peuvent être utilisés s'ils sont nécessaires pour votre flux de travail. Consultez la description du [HistoryEvent](#) cours pour plus d'informations sur chaque type d'événement.

+ REMARQUE : Les chaînes dans `switch` les instructions ont été introduites dans Java 7. Si vous utilisez une version antérieure de Java, vous pouvez utiliser la [EventType](#) classe pour convertir la valeur `String` renvoyée par `event.getType()` d'énumération, puis de nouveau en une `String` si nécessaire :

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

1. Après l'instruction `switch`, ajoutez du code pour répondre avec une décision appropriée en fonction de la tâche qui a été reçue.

```
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);
```

- Si l'activité n'a pas encore été planifiée, nous répondons par une `ScheduleActivityTask` décision, qui fournit des informations sous forme de [ScheduleActivityTaskDecisionAttributes](#) structure sur l'activité à planifier ensuite, y compris toutes les données qui Amazon SWF doivent être envoyées à l'activité. Amazon SWF
- Si l'activité est terminée, nous considérons que l'ensemble du flux de travail est terminé et nous répondons par une `CompletedWorkflowExecution` décision, en remplissant une [CompleteWorkflowExecutionDecisionAttributes](#) structure fournissant des détails sur le flux de travail terminé. Dans ce cas, nous renvoyons le résultat de l'activité.

Dans les deux cas, les informations de décision sont ajoutées à la liste `Decision` qui a été déclarée en haut de la méthode.

2. Terminez la tâche de décision en renvoyant la liste des objets `Decision` collectés pendant le traitement de la tâche. Ajoutez ce code à la fin de la méthode `executeDecisionTask` que nous avons écrite :

```
swf.respondDecisionTaskCompleted(
    new RespondDecisionTaskCompletedRequest()
        .withTaskToken(taskToken)
        .withDecisions(decisions));
```

La méthode `respondDecisionTaskCompleted` du client SWF prend le jeton de tâche qui identifie la tâche, ainsi que la liste d'objets `Decision`.

Implémentation du démarreur de flux de travail

Pour finir, nous allons écrire du code pour démarrer l'exécution du flux de travail.

1. Ouvrez votre éditeur de texte et créez le fichier `WorkflowStarter.java`, en ajoutant une déclaration de package et des déclarations d'importation conformément aux [étapes courantes](#).
2. Ajoutez la 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() + "'.");  
    }  
}
```

La classe `WorkflowStarter` est constituée d'une seule méthode, `main`, qui prend un argument facultatif transmis dans la ligne de commande en tant que données d'entrée pour le flux de travail.

La méthode client `SWFstartWorkflowExecution`, prend un [StartWorkflowExecutionRequest](#) objet en entrée. Ici, en plus de spécifier le domaine et le type de flux de travail à exécuter, nous lui fournissons :

- un nom d'exécution de flux de travail lisible par l'utilisateur ;
- des données d'entrée de flux de travail (fournies dans la ligne de commande dans notre exemple) ;
- une valeur de délai d'expiration qui représente la durée, en secondes, que l'exécution de la totalité du flux de travail doit respecter.

L'objet [Run](#) renvoyé par `startWorkflowExecution` fournit un ID d'exécution, c'est-à-dire une valeur qui peut être utilisée pour identifier cette exécution de flux de travail particulière dans l'historique Amazon SWF de vos exécutions de flux de travail.

+ REMARQUE : L'ID d'exécution est généré par Amazon SWF le nom d'exécution du flux de travail que vous transmettez au démarrage de l'exécution du flux de travail et n'est pas le même que celui-ci.

Génération de l'exemple

Pour générer l'exemple de projet avec Maven, accédez au répertoire `helloswf` et tapez :

```
mvn package
```

Le fichier `helloswf-1.0.jar` résultant est généré dans le répertoire `target`.

Exécution de l'exemple

L'exemple est constitué de quatre classes exécutables distinctes qui sont exécutées indépendamment les unes des autres.

Note

Si vous utilisez un système Linux, macOS ou Unix, vous pouvez tous les exécuter, l'un après l'autre, dans une seule fenêtre de terminal. Si vous exécutez Windows, vous devez ouvrir deux instances de ligne de commande supplémentaires et accéder au répertoire `helloworld` dans chacune d'entre elles.

Définition du chemin de classe Java

Bien que Maven ait géré les dépendances pour vous, pour exécuter l'exemple, vous devez fournir la bibliothèque du AWS SDK et ses dépendances sur votre chemin de classe Java. Vous pouvez définir la variable d'environnement `CLASSPATH` en fonction de l'emplacement des bibliothèques de votre AWS SDK et du `third-party/lib` répertoire du SDK, qui inclut les dépendances nécessaires :

```
export CLASSPATH='target/helloworld-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/*'
java example.swf.hello.HelloTypes
```

ou utilisez l'option `-cp` de la `java` commande pour définir le chemin de classe lors de l'exécution de chaque application.

```
java -cp target/helloworld-1.0.jar:/path/to/sdk/lib/*:/path/to/sdk/third-party/lib/* \
example.swf.hello.HelloTypes
```

C'est à vous de décider ce que vous souhaitez utiliser. Si vous n'avez eu aucun problème à créer le code, essayez d'exécuter les exemples et obtenez une série d'erreurs « `NoClassDefFound` ». Cela est probablement dû au fait que le chemin de classe n'est pas correctement défini.

Enregistrement du domaine, et des types de flux de travail et d'activité

Avant d'exécuter vos travaux et le démarreur de flux de travail, vous devez enregistrer le domaine, ainsi que vos types de flux de travail et d'activité. Le code pour ce faire a été implémenté dans le [flux de travail et les types d'activité Enregistrer un domaine](#).

Après la génération, et si vous avez [défini CLASSPATH](#), vous pouvez exécuter le code d'enregistrement en lançant la commande :

```
echo 'Supply the name of one of the example classes as an argument.'
```

Démarrage des travaux d'activité et de flux de travail

Maintenant que les types ont été enregistrés, vous pouvez démarrer les travaux d'activité et de flux de travail. Ils continueront à s'exécuter et à rechercher des tâches jusqu'à ce qu'ils soient supprimés. Vous devez donc les exécuter dans des fenêtres de terminal distinctes ou, si vous utilisez Linux, macOS ou Unix, vous pouvez utiliser l'opérateur pour que chacun d'entre eux génère un processus distinct lors de son exécution.

```
echo 'If there are arguments to the class, put them in quotes after the class
name.'
exit 1
```

Si vous exécutez ces commandes dans des fenêtres distinctes, omettez l'opérateur & final dans chaque ligne.

Démarrage de l'exécution de flux de travail

Maintenant que vos travaux d'activité et de flux de travail exécutent l'interrogation, vous pouvez démarrer l'exécution du flux de travail. Ce processus s'exécute jusqu'à ce que le flux de travail renvoie un état terminé. Vous devez l'exécuter dans une nouvelle fenêtre de terminal (sauf si vous avez exécuté vos travaux en tant que nouveaux processus générés à l'aide de l'opérateur &).

```
fi
```

Note

Si vous souhaitez fournir vos propres données d'entrée, qui seront transmises d'abord au flux de travail, puis à l'activité, ajoutez-les à la ligne de commande. Par exemple :

```
echo "## Running $className..."
```

Une fois que vous avez démarré l'exécution du flux de travail, vous devez commencer à voir la sortie fournie par les deux travaux et par l'exécution de flux de travail proprement dite. Lorsque le flux de travail est terminé, sa sortie est affichée à l'écran.

Exécution de la source pour cet exemple

Vous pouvez parcourir la [source complète](#) de cet exemple sur Github dans le [aws-java-developer-guide](#) référentiel.

Pour plus d'informations

- Les travaux présentés ici peuvent entraîner la perte des tâches s'ils sont fermés alors que l'interrogation du flux de travail est encore en cours. Pour découvrir comment fermer correctement les travaux, consultez [Arrêt normal des travaux d'activité et de flux de travail](#).
- Pour en savoir plus Amazon SWF, rendez-vous sur la page d'[Amazon SWF](#) accueil ou consultez le [Guide du Amazon SWF développeur](#).
- Vous pouvez utiliser AWS Flow Framework pour Java pour écrire des flux de travail plus complexes dans un style Java élégant à l'aide d'annotations. [Pour de plus amples informations, veuillez consulter consulter consulter consulter consulter consulter consulter consulter consulter consulter consulter consulter consulter consulter consulter consulter AWS Flow Framework](#)

Tâches Lambda

À la place d'activités Amazon SWF ou conjointement à celles-ci, vous pouvez utiliser des fonctions [Lambda](#) pour représenter des unités de travail de votre flux de travail et planifier ces unités de la même manière que les activités.

Cette rubrique est consacrée à la façon d'implémenter Amazon SWF Lambda tâches utilisant le AWS SDK for Java. Pour plus d'informations sur Lambda tâches en général, voir [AWS Lambda Tâches](#) dans le Amazon SWF Manuel du développeur

Configuration d'un rôle IAM inter-services pour exécuter votre fonction Lambda

Avant Amazon SWF peut exécuter votre Lambda, vous devez configurer un rôle IAM pour donner Amazon SWF autorisation d'exécution Lambda fonctionne en votre nom. Pour plus d'informations sur la procédure à utiliser, consultez [AWS Lambda Tâches](#).

Vous avez besoin de l'Amazon Resource Name (ARN) de ce rôle IAM lorsque vous enregistrez un flux de travail qui utilisera Lambda tâches.

Créer une fonction Lambda

Vous pouvez écrire des fonctions Lambda dans plusieurs langages, y compris Java. Pour plus d'informations sur la création, le déploiement et l'utilisation Lambda, consultez le [AWS Lambda Manuel du développeur](#).

Note

Peu importe la langue que vous utilisez pour écrire votre Lambda, il peut être programmé et exécuté par n'importe quel Amazon SWF, quelle que soit la langue dans laquelle votre code de flux de travail est écrit. Amazon SWF gère les détails de l'exécution de la fonction, ainsi que la transmission des données vers et depuis celle-ci.

Voici un jeu simple Lambda qui peut être utilisée au lieu de l'activité dans [Création d'une solution simple Amazon SWF Application](#).

- Cette version est écrit en JavaScript, qui peut être saisie directement à l'aide du kit [AWS Management Console](#) :

```
exports.handler = function(event, context) {
    context.succeed("Hello, " + event.who + "!");
};
```

- Voici la même fonction écrit en Java, que vous pouvez également déployer et exécuter sur 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

Pour en savoir plus sur le déploiement des fonctions Java sur Lambda, consultez [Création d'un package de déploiement \(Java\)](#) dans le AWS Lambda Manuel du développeur. Vous pouvez également consulter la section intitulée [Modèle de programmation pour la création Lambda Fonctions en Java](#).

Les fonctions Lambda prennent un objet event ou input comme premier paramètre et un objet context comme deuxième paramètre, lequel fournit des informations sur la demande d'exécution de la fonction Lambda. Cette fonction particulière attend une entrée au format JSON, avec un champ who défini sur le nom utilisé pour créer la salutation.

Enregistrement d'un flux de travail à utiliser avec Lambda

Pour qu'un flux de travail planifie une fonction Lambda, vous devez fournir le nom du rôle IAM qui autorise Amazon SWF à appeler des fonctions Lambda. Vous pouvez définir celui-ci pendant l'enregistrement du flux de travail à l'aide des méthodes `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) {
```

Planification d'une tâche Lambda

La planification d'une tâche Lambda est similaire à la planification d'une activité. Vous fournissez un [La décision](#) avec une fonction « ScheduleLambda Function » [DecisionType](#) et avec [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(
```

Dans la classe `ScheduleLambdaFunctionDecisionAttributes`, vous devez fournir un nom (name), qui est l'ARN de la fonction Lambda à appeler et un id, qui est le nom utilisé par Amazon SWF pour identifier la fonction Lambda dans les journaux d'historique.

Vous pouvez également fournir une entrée (input) facultative pour la fonction Lambda et définir sa valeur start to close timeout, qui est le nombre de secondes pendant lesquelles la fonction Lambda est autorisée à s'exécuter avant de générer un événement `LambdaFunctionTimedOut`.

Note

Ce code utilise la classe [AWSLambdaClient](#) pour récupérer l'ARN de la fonction Lambda, en fonction du nom de celle-ci. Vous pouvez utiliser cette technique pour éviter de coder en dur l'ARN complet (qui inclut votre Compte AWSID) dans votre code.

Gestion des événements de fonction Lambda dans votre décideur

Les tâches Lambda génèrent différents événements sur lesquels vous pouvez agir lorsque vous recherchez des tâches de décision dans votre travail de flux de travail, correspondant au cycle de vie de votre tâche Lambda, avec pour [EventType](#) des valeurs telles que `LambdaFunctionScheduled`, `LambdaFunctionStarted` et `LambdaFunctionCompleted`. Si la fonction Lambda échoue, ou s'exécute plus longtemps que sa valeur de délai d'expiration définie, vous recevez un type d'événement `LambdaFunctionFailed` ou `LambdaFunctionTimedOut`, respectivement.

```
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;
```

Réception de la sortie de votre fonction Lambda

Lorsque vous recevez un `LambdaFunctionCompleted` [`EventType`](#), you can retrieve your `0` function's return value by first calling ``getLambdaFunctionCompletedEventAttributes`` sur le [`HistoryEvent`](#) pour obtenir un [`LambdaFunctionCompletedEventAttributes`](#), puis en appelant `son`getResult`` pour récupérer la sortie de la `laLambda`:

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

Exécution de la source pour cet exemple

Vous pouvez parcourir la source complète [awsdocs/aws-java-developer-guide/tree/master/doc_source/snippets/helloswf_lambda/](https://awsdocs.aws.amazon.com/lambda/latest/dg/developing-applications/hello-swf-lambda.html) de cet exemple sur Github dans le référentiel `aws-java-developer-guide`.

Arrêt normal des travaux d'activité et de flux de travail

Le [`Création d'une application simple Amazon SWF Application`](#) fournit l'implémentation complète d'une application de flux de travail simple, composée d'une application d'inscription, d'un travail d'activité et d'un travail de flux de travail ainsi que d'un démarreur de flux de travail.

Les classes de travaux sont conçues pour s'exécuter continuellement, via l'interrogation des tâches envoyées par Amazon SWF, afin d'exécuter des activités ou de retourner des décisions. Une fois qu'une demande d'interrogation a été faite, Amazon SWF enregistre l'interrogateur et tente de lui attribuer une tâche.

Si le travail de flux de travail se termine pendant une interrogation longue, Amazon SWF peut continuer à essayer d'envoyer une tâche au travail terminé, ce qui se traduit par une perte de tâche (jusqu'à ce que la tâche expire).

Un moyen de gérer cette situation consiste à attendre que toutes les demandes d'interrogation longue envoient un retour avant que le travail ne se termine.

Dans cette rubrique, nous allons réécrire le travail d'activité depuis `helloswf`, à l'aide des hooks d'arrêt de Java afin de tenter un arrêt approprié du travail d'activité.

Voici le code complet :

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

Dans cette version, le code d'interrogation qui était dans la fonction `main` de la version d'origine a été déplacé dans sa propre méthode `pollAndExecute`.

La fonction `main` utilise désormais un [CountDownLatch](#) conjointement avec un [hook d'arrêt](#) pour que le thread attende 60 secondes au plus après la demande de sa résiliation et avant de permettre l'arrêt du thread.

Enregistrement de domaines

Chaque flux de travail et chaque activité dans [Amazon SWF](#) NécesSITE domaine pour courir dans.

1. Créez un nouvel objet [RegisterDomainRequest](#), en lui fournissant au moins le nom du domaine et la période de conservation pour les exécutions de flux de travail (ces paramètres sont tous deux obligatoires).
2. Appelez la méthode [AmazonSimpleWorkflowClient.registerDomain](#) avec l'objet `RegisterDomainRequest`.
3. Interceptez l'exception [DomainAlreadyExistsException](#) si le domaine que vous demandez existe déjà (dans ce cas, aucune action n'est généralement requise).

Le code suivant illustre la procédure :

```
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!");
}
}
```

Affichage des domaines

Vous pouvez lister les [Amazon SWF](#) Répertoire les domaines associés à votre compte et AWS région par type d'enregistrement.

1. Création d'un [ListDomainsRequest](#) et spécifiez l'état d'enregistrement des domaines qui vous intéressent (obligatoire).
2. Appelez [AmazonSimpleWorkflowClient.listDomains](#) avec l'objet ListDomainRequest. Les résultats sont fournis dans un objet [DomainInfos](#).
3. Appelez [getDomainInfos](#) sur l'objet renvoyé pour obtenir la liste des objets [DomainInfo](#).
4. Appelez [getName](#) sur chaque objet DomainInfo pour obtenir son nom.

Le code suivant illustre la procédure :

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

Exemples de code inclus avec le kit SDK

Le kit AWS SDK for Java est fourni avec des exemples de code ; ceux-ci illustrent un grand nombre des fonctions du kit SDK au sein de programmes qui peuvent être générés et exécutés. Vous pouvez étudier ou modifier ces exemples pour implémenter vos propres exemples AWS solutions utilisant le kit AWS SDK for Java.

Comment obtenir les exemples

Les exemples de code du kit AWS SDK for Java sont disponibles dans le répertoire samples du kit SDK. Si vous avez téléchargé et installé le SDK à l'aide des informations de [Configuration de l'AWS SDK for Java](#), vous avez déjà les échantillons sur votre système.

Vous pouvez également consulter les exemples les plus récents sur le référentiel GitHub du kit AWS SDK for Java, dans le répertoire [src/samples](#).

Génération et exécution d'exemples à l'aide de la ligne de commande

Les exemples incluent les scripts de génération [Ant](#) afin que vous puissiez facilement les générer et les exécuter à partir de la ligne de commande. Chaque exemple contient aussi un fichier README au format HTML, qui comporte des informations propres à chaque exemple.

Note

Si vous parcourez l'exemple de code sur GitHub, cliquez sur le bouton Raw dans la vue du code source lorsque vous consultez le fichier README.html de l'exemple. En mode brut, le code HTML s'affiche comme prévu dans votre navigateur.

Prérequis

Avant de lancer l'un des AWS SDK for Java échantillons, vous devez définir votre AWS informations d'identification dans l'environnement ou avec le kit AWS CLI, tel que spécifié dans [Configuration AWS Informations d'identification et région pour le développement](#). Les exemples utilisent le fournisseur d'informations d'identification par défaut chaque fois que possible. En définissant ainsi vos informations d'identification, vous évitez la pratique risquée d'insérer vos informations d'identification AWS Les informations d'identification dans les fichiers du répertoire du code source (où elles peuvent être malencontreusement enregistrées et partagées publiquement).

Exécution des exemples

1. Accédez au répertoire contenant l'exemple de code. Par exemple, si vous êtes dans le répertoire racine du kit AWS Téléchargez le kit SDK et souhaitez exécuter le `AwsConsoleAppExemple`, vous devez taper la commande suivante :

```
cd samples/AwsConsoleApp
```

2. Générez et exécutez l'exemple avec Ant. Comme, par défaut, la cible de génération exécute les deux actions, il vous suffit d'entrer l'instruction suivante :

```
ant
```

L'exemple affiche les informations sur la sortie standard, par exemple :

```
=====
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.
```

Génération et exécution des exemples à l'aide de l'IDE Eclipse

Si vous utilisez AWS Toolkit for Eclipse, vous pouvez également démarrer un nouveau projet dans Eclipse basé sur le kit AWS SDK for Java ou ajouter le kit SDK à un projet Java existant.

Prérequis

Après avoir installé AWS Toolkit for Eclipse, nous vous recommandons de configurer la boîte à outils avec vos informations d'identification de sécurité. Vous pouvez effectuer cette opération à tout moment en choisissant **Préférences** à partir des **Fenêtres** dans Eclipse, puis choisissez la commande **AWS Boîte à outils** Section.

Exécution des exemples

1. Ouvrez Eclipse.
2. Création d'un nouveau **AWS Projet Java**. Dans Eclipse, dans le menu **File (Fichier)**, choisissez **New (Nouveau)**, puis cliquez sur **Project (Projet)**. L'Assistant **New Project (Nouveau projet)** s'ouvre.
3. Développez le kit **AWS** catégorie, puis choisissez **AWS Projet Java**.
4. Choisissez **Next (Suivant)**. La page des paramètres du projet s'affiche.

5. Entrez un nom dans la zone Project Name (Nom du projet). LeAWS SDK for JavaLe groupe d'exemples affiche les exemples disponibles dans le kit, comme décrit précédemment.
6. Sélectionnez les exemples que vous voulez inclure dans votre projet en cochant chaque case correspondante.
7. Saisissez votreAWSInformations d'identification . Si vous avez déjà configuré la boîte à outils AWS Toolkit for Eclipse avec vos informations d'identification, celles-ci sont automatiquement renseignées.
8. Choisissez Finish (Terminer). Le projet est créé et ajouté au Project Explorer (Explorateur de projet).
9. Sélectionnez l'exemple de fichier .j ava que vous voulez exécuter. Ainsi, pour l'exemple Amazon S3, choisissez S3Sample.j ava.
- 10.Choisissez Run (Exécuter) dans le menu Run (Exécuter).
- 11.Cliquez avec le bouton droit sur le projet dans Project Explorer (Explorateur de projet), pointez vers Build Path (Chemin de génération), puis choisissez Add Libraries (Ajouter les bibliothèques).
- 12.Choisissez AWSSDK Java, choisissezSuivant, puis suivez les instructions à l'écran.

Sécurité pour AWS SDK for Java

Chez Amazon Web Services (AWS), la sécurité dans le cloud est la priorité principale. En tant que client AWS, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des organisations les plus pointilleuses sur la sécurité. La sécurité est une responsabilité partagée entre vous AWS et vous. Le [modèle de responsabilité partagée](#) décrit cela comme la sécurité du cloud et la sécurité dans le cloud.

Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute tous les services proposés dans le AWS cloud et de vous fournir des services que vous pouvez utiliser en toute sécurité. Notre responsabilité en matière de sécurité est notre priorité absolue AWS, et l'efficacité de notre sécurité est régulièrement testée et vérifiée par des auditeurs tiers dans le cadre des [programmes de AWS conformité](#).

Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez et par d'autres facteurs, notamment la sensibilité de vos données, les exigences de votre organisation et les lois et réglementations applicables.

Ce AWS produit ou service suit le [modèle de responsabilité partagée](#) par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la [AWS page de documentation sur la sécurité AWS des services et les services concernés par les efforts de AWS conformité par programme de conformité](#).

Rubriques

- [Protection des données dans la version AWS SDK for Java 1.x](#)
- [AWS SDK for Java support pour TLS](#)
- [Gestion de l'identité et des accès](#)
- [Validation de conformité pour ce AWS produit ou service](#)
- [Résilience pour ce AWS produit ou service](#)
- [Sécurité de l'infrastructure pour ce AWS produit ou service](#)
- [Amazon S3 Migration du client de chiffrement](#)

Protection des données dans la version AWS SDK for Java 1.x

Le [modèle de responsabilité partagée](#) s'applique à la protection des données dans ce AWS produit ou service. Comme décrit dans ce modèle, AWS est responsable de la protection de l'infrastructure

mondiale qui gère l'ensemble du AWS cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure est de votre responsabilité. Ce contenu comprend les tâches de configuration et de gestion de la sécurité des services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez les [FAQ sur la confidentialité des données](#). Pour plus d'informations sur la protection des données en Europe, consultez le [modèle de responsabilité AWS partagée et le billet de blog sur le RGPD](#) sur le blog sur la AWS sécurité.

Pour des raisons de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer des comptes utilisateur individuels avec AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, avec tous les contrôles de sécurité par défaut au sein AWS des services.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui vous aident à découvrir et à sécuriser les données personnelles qui y sont stockées. Amazon S3
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez [Federal Information Processing Standard \(FIPS\) 140-2](#) (Normes de traitement de l'information fédérale).

Nous vous recommandons vivement de ne jamais placer d'informations identifiables sensibles, telles que les numéros de compte de vos clients, dans des champs de formulaire comme Name (Nom). Cela inclut lorsque vous travaillez avec ce AWS produit ou service ou avec d'autres AWS services à l'aide de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous entrez dans ce AWS produit ou service ou dans d'autres services peuvent être récupérées pour être incluses dans les journaux de diagnostic. Lorsque vous fournissez une URL à un serveur externe, n'incluez pas les informations d'identification non chiffrées dans l'URL pour valider votre demande adressée au serveur.

AWS SDK for Java support pour TLS

Les informations suivantes s'appliquent uniquement à l'implémentation Java SSL (l'implémentation SSL par défaut dans le AWS SDK for Java). Si vous utilisez une implémentation de SSL différente, consultez votre implémentation de SSL spécifique pour savoir comment appliquer les versions de TLS.

Comment vérifier la version de TLS

Consultez la documentation de votre fournisseur de machine virtuelle Java (JVM) pour déterminer quelles versions de TLS sont prises en charge sur votre plate-forme. Pour certaines machines virtuelles virtuelles, le code suivant indiquera les versions SSL prises en charge.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters()).getProto
```

Pour voir la liaison SSL en action et quelle version de TLS est utilisée, vous pouvez utiliser la propriété système `javax.net.debug`.

```
java app.jar -Djavax.net.debug=ssl
```

Note

TLS 1.3 est incompatible avec les versions 1.9.5 à 1.10.31 du SDK for Java. Pour plus d'informations, consultez le billet de blog suivant.

<https://aws.amazon.com/blogs/developer/tls-1-3-incompatibility-with-aws-sdk-for-java-versions-1-9-5-to-1-10-31/>

Application d'une version minimale de TLS

Le SDK préfère toujours la dernière version TLS prise en charge par la plateforme et le service. Si vous souhaitez appliquer une version minimale de TLS spécifique, consultez la documentation de votre machine virtuelle Java. Pour les machines virtuelles Java basées sur OpenJDK, vous pouvez utiliser la propriété système `jdk.tls.client.protocols`

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

Consultez la documentation de votre machine virtuelle Java pour connaître les valeurs prises en charge par PROTOCOLS.

Gestion de l'identité et des accès

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser AWS les ressources. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- [Public ciblé](#)
- [Authentification par des identités](#)
- [Gestion des accès à l'aide de politiques](#)
- [Comment Services AWS travailler avec IAM](#)
- [Résolution des problèmes AWS d'identité et d'accès](#)

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez. AWS

Utilisateur du service : si vous avez l' Services AWS habitude de faire votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de nouvelles AWS fonctionnalités pour effectuer votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans AWS, consultez [Résolution des problèmes AWS d'identité et d'accès](#) le guide de l'utilisateur du Service AWS que vous utilisez.

Administrateur du service — Si vous êtes responsable des AWS ressources de votre entreprise, vous avez probablement un accès complet à AWS. C'est à vous de déterminer les AWS fonctionnalités et les ressources auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM avec AWS, consultez le guide de l'utilisateur Service AWS que vous utilisez.

Administrateur IAM – Si vous êtes un administrateur IAM, vous souhaitez peut-être en savoir plus sur la façon d'écrire des politiques pour gérer l'accès à AWS. Pour consulter des exemples de politiques AWS basées sur l'identité que vous pouvez utiliser dans IAM, consultez le guide de l'utilisateur Service AWS que vous utilisez.

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section [Comment vous connecter à votre compte Compte AWS dans](#) le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section [Signature des demandes AWS d'API](#) dans le guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, consultez [Authentification multifactorielle](#) dans le Guide de l'utilisateur AWS IAM Identity Center et [Utilisation de l'authentification multifactorielle \(MFA\) dans l'interface AWS](#) dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur root, consultez [Tâches nécessitant des informations d'identification d'utilisateur root](#) dans le Guide de l'utilisateur IAM.

Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède à l'aide des informations d'identification fournies Services AWS par le biais d'une source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous pouvez vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. Pour obtenir des informations sur IAM Identity Center, consultez [Qu'est-ce que IAM Identity Center ?](#) dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un [utilisateur IAM](#) est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès.

Pour plus d'informations, consultez [Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification](#) dans le Guide de l'utilisateur IAM.

Un [groupe IAM](#) est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez [Quand créer un utilisateur IAM \(au lieu d'un rôle\)](#) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un [rôle IAM](#) est une identité au sein de votre Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console [changeant de rôle](#). Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez [Utilisation de rôles IAM](#) dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez la rubrique [Jeux d'autorisations](#) dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent

le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez [Différence entre les rôles IAM et les politiques basées sur les ressources](#) dans le Guide de l'utilisateur IAM.

- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
- Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez [Sessions de transmission d'accès](#).
- Rôle de service : il s'agit d'un [rôle IAM](#) attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez [Création d'un rôle pour la délégation d'autorisations à un Service AWS](#) dans le Guide de l'utilisateur IAM.
- Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une instance EC2 et qui envoient des demandes d'API. AWS CLI AWS Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un AWS rôle à une instance EC2 et le mettre à la disposition de toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations,

consultez [Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2](#) dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez [Quand créer un rôle IAM \(au lieu d'un utilisateur\)](#) dans le Guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez [Vue d'ensemble des politiques JSON](#) dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Création de politiques IAM](#) dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez [Choix entre les politiques gérées et les politiques en ligne](#) dans le Guide de l'utilisateur IAM.

politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d'accès, consultez [Vue d'ensemble des listes de contrôle d'accès \(ACL\)](#) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- **Limite d'autorisations** : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez [Limites d'autorisations pour des entités IAM](#) dans le Guide de l'utilisateur IAM.
- **Politiques de contrôle des services (SCP)** — Les SCP sont des politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer les politiques de contrôle des services (SCP) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les organisations et les SCP, consultez [Fonctionnement des SCP](#) dans le Guide de l'utilisateur AWS Organizations .
- **Politiques de séance** : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez [politiques de séance](#) dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS détermine s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section [Logique d'évaluation des politiques](#) dans le guide de l'utilisateur IAM.

Comment Services AWS travailler avec IAM

Pour obtenir une vue d'ensemble du Services AWS fonctionnement de la plupart des fonctionnalités IAM, consultez les [AWS services compatibles avec IAM](#) dans le guide de l'utilisateur IAM.

Pour savoir comment utiliser un service spécifique Service AWS avec IAM, consultez la section relative à la sécurité du guide de l'utilisateur du service concerné.

Résolution des problèmes AWS d'identité et d'accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS IAM.

Rubriques

- [Je ne suis pas autorisé à effectuer une action dans AWS](#)
- [Je ne suis pas autorisé à effectuer iam : PassRole](#)
- [Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes AWS ressources](#)

Je ne suis pas autorisé à effectuer une action dans AWS

Si vous recevez une erreur qui indique que vous n'êtes pas autorisé à effectuer une action, vos politiques doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource `my-example-widget` fictive, mais ne dispose pas des autorisations `aws:GetWidget` fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

Dans ce cas, la politique qui s'applique à l'utilisateur `mateojackson` doit être mise à jour pour autoriser l'accès à la ressource `my-example-widget` à l'aide de l'action `aws:GetWidget`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter `iam:PassRole` l'action, vos stratégies doivent être mises à jour afin de vous permettre de transmettre un rôle à AWS.

Certains vos Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour exécuter une action dans AWS. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes AWS ressources

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si ces fonctionnalités sont prises AWS en charge, consultez [Comment Services AWS travailler avec IAM](#).
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section [Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez](#) dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section [Fournir un accès à des ressources Comptes AWS détenues par des tiers](#) dans le guide de l'utilisateur IAM.

- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez [Fournir un accès à des utilisateurs authentifiés en externe \(fédération d'identité\)](#) dans le Guide de l'utilisateur IAM.
- Pour découvrir quelle est la différence entre l'utilisation des rôles et l'utilisation des politiques basées sur les ressources pour l'accès entre comptes, consultez [Différence entre les rôles IAM et les politiques basées sur les ressources](#) dans le Guide de l'utilisateur IAM.

Validation de conformité pour ce AWS produit ou service

Pour savoir si un [programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité](#) spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de [AWS conformité Programmes AWS](#) de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir [Téléchargement de rapports dans AWS Artifact](#) .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- [Guides de démarrage rapide sur la sécurité et la conformité](#) : ces guides de déploiement abordent les considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de base axés sur AWS la sécurité et la conformité.
- [Architecture axée sur la sécurité et la conformité HIPAA sur Amazon Web Services](#) : ce livre blanc décrit comment les entreprises peuvent créer des applications AWS conformes à la loi HIPAA.

Note

Tous ne Services AWS sont pas éligibles à la loi HIPAA. Pour plus d'informations, consultez le [HIPAA Eligible Services Reference](#).

- AWS Ressources de <https://aws.amazon.com/compliance/resources/> de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- [AWS Guides de conformité destinés aux clients](#) — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière

de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).

- [Évaluation des ressources à l'aide des règles](#) du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- [AWS Security Hub](#)— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez [Référence des contrôles Security Hub](#).
- [Amazon GuardDuty](#) — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- [AWS Audit Manager](#)— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Ce AWS produit ou service suit le [modèle de responsabilité partagée](#) par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la [AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité](#).

Résilience pour ce AWS produit ou service

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité.

Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant.

Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de

disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section [Infrastructure AWS mondiale](#).

Ce AWS produit ou service suit le [modèle de responsabilité partagée](#) par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la [AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité](#).

Sécurité de l'infrastructure pour ce AWS produit ou service

Ce AWS produit ou service utilise des services gérés et est donc protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section [Sécurité du AWS cloud](#). Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section [Protection de l'infrastructure](#) dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à ce AWS produit ou service via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser [AWS Security Token Service](#) (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Ce AWS produit ou service suit le [modèle de responsabilité partagée](#) par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la [AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité](#).

Amazon S3 Migration du client de chiffrement

Cette rubrique explique comment migrer vos applications de la version 1 (V1) du client de chiffrement Amazon Simple Storage Service (Amazon S3) vers la version 2 (V2) et garantir la disponibilité des applications tout au long du processus de migration.

Prérequis

Amazon S3 le chiffrement côté client nécessite les éléments suivants :

- Java 8 ou version ultérieure installé dans votre environnement d'application. [Il AWS SDK for Java fonctionne avec le kit de développement Oracle Java SE et avec les distributions du kit de développement Open Java \(OpenJDK\) Amazon Correttotelles que Red Hat OpenJDK et JDK. AdoptOpen](#)
- Le [package Bouncy Castle Crypto](#). Vous pouvez placer le fichier .jar de Bouncy Castle sur le chemin de classe de votre environnement d'application ou ajouter une dépendance à l'ArtifactID `bcprov-ext-jdk15on` (avec le groupID de) à votre fichier Maven. `org.bouncycastle` `pom.xml`

Présentation de la migration

Cette migration s'effectue en deux phases :

1. Mettez à jour les clients existants pour lire les nouveaux formats. Mettez à jour votre application pour utiliser la version 1.11.837 ou ultérieure AWS SDK for Java et redéployez l'application. Cela permet aux Amazon S3 clients du service de chiffrement côté client de votre application de déchiffrer les objets créés par les clients du service V2. Si votre application utilise plusieurs AWS SDK, vous devez mettre à jour chaque SDK séparément.
2. Migrez les clients de chiffrement et de déchiffrement vers la version V2. Une fois que tous vos clients de chiffrement V1 peuvent lire les formats de chiffrement V2, mettez à jour les Amazon S3 clients de chiffrement et de déchiffrement côté client dans le code de votre application pour utiliser leurs équivalents V2.

Mettre à jour les clients existants pour lire les nouveaux formats

Le client de chiffrement V2 utilise des algorithmes de chiffrement que les anciennes versions AWS SDK for Java ne prennent pas en charge.

La première étape de la migration consiste à mettre à jour vos clients de chiffrement V1 afin qu'ils utilisent la version 1.11.837 ou ultérieure du. AWS SDK for Java(Nous vous recommandons de passer à la dernière version, que vous trouverez dans la [version 1.x de référence de l'API Java.](#)) Pour ce faire, mettez à jour la dépendance dans la configuration de votre projet. Une fois la configuration de votre projet mise à jour, reconstruisez votre projet et redéployez-le.

Une fois ces étapes terminées, les clients de chiffrement V1 de votre application pourront lire les objets écrits par les clients de chiffrement V2.

Mettre à jour la dépendance dans la configuration de votre projet

Modifiez le fichier de configuration de votre projet (par exemple, pom.xml ou build.gradle) pour utiliser la version 1.11.837 ou ultérieure du. AWS SDK for Java Reconstruisez ensuite votre projet et redéployez-le.

L'exécution de cette étape avant le déploiement du nouveau code d'application permet de garantir la cohérence des opérations de chiffrement et de déchiffrement au sein de votre flotte pendant le processus de migration.

Exemple d'utilisation de Maven

Extrait d'un fichier 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>
```

Exemple d'utilisation de Gradle

Extrait d'un fichier build.gradle :

```
dependencies {
  implementation platform('com.amazonaws:aws-java-sdk-bom:1.11.837')
```

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

Migrer les clients de chiffrement et de déchiffrement vers la version V2

Une fois que votre projet a été mis à jour avec la dernière version du SDK, vous pouvez modifier le code de votre application pour utiliser le client V2. Pour ce faire, commencez par mettre à jour votre code afin d'utiliser le nouveau générateur de clients de services. Fournissez ensuite du matériel de chiffrement à l'aide d'une méthode du générateur qui a été renommée, et configurez davantage votre client de service selon les besoins.

Ces extraits de code montrent comment utiliser le chiffrement côté client avec le et fournissent des comparaisons entre AWS SDK for Java les clients de chiffrement V1 et V2.

V1

```
// minimal configuration in V1; default CryptoMode.EncryptionOnly.  
EncryptionMaterialsProvider encryptionMaterialsProvider = ...  
AmazonS3Encryption encryptionClient = AmazonS3EncryptionClient.encryptionBuilder()  
    .withEncryptionMaterials(encryptionMaterialsProvider)  
    .build();
```

V2

```
// minimal configuration in V2; default CryptoMode.StrictAuthenticatedEncryption.  
EncryptionMaterialsProvider encryptionMaterialsProvider = ...  
AmazonS3EncryptionV2 encryptionClient = AmazonS3EncryptionClientV2.encryptionBuilder()  
    .withEncryptionMaterialsProvider(encryptionMaterialsProvider)  
    .withCryptoConfiguration(new CryptoConfigurationV2()  
        // The following setting allows the client to read V1  
        // encrypted objects  
        .withCryptoMode(CryptoMode.AuthenticatedEncryption)  
    )  
    .build();
```

L'exemple ci-dessus définit la valeur `cryptoMode` à `AuthenticatedEncryption`. Il s'agit d'un paramètre qui permet à un client de chiffrement V2 de lire des objets écrits par un client de chiffrement V1. Si votre client n'a pas besoin de pouvoir lire des objets écrits par un client V1, nous vous recommandons d'utiliser `StrictAuthenticatedEncryption` plutôt le paramètre par défaut de.

Création d'un client de chiffrement V2

Le client de chiffrement V2 peut être créé en appelant `AmazonS3 EncryptionClient v2.encryptionBuilder ()`.

Vous pouvez remplacer tous vos clients de chiffrement V1 existants par des clients de chiffrement V2. Un client de chiffrement V2 sera toujours en mesure de lire tout objet écrit par un client de chiffrement V1 tant que vous l'autorisez à le faire en configurant le client de chiffrement V2 pour qu'il utilise le `AuthenticatedEncryption`cryptoMode`.

La création d'un nouveau client de chiffrement V2 est très similaire à la création d'un client de chiffrement V1. Toutefois, il existe quelques différences :

- Vous utiliserez un `CryptoConfigurationV2` objet pour configurer le client au lieu d'un `CryptoConfiguration` objet. Ce paramètre est obligatoire.
- Le `cryptoMode` paramètre par défaut pour le client de chiffrement V2 est `StrictAuthenticatedEncryption`. Pour le client de chiffrement V1, c'est le `EncryptionOnly`.
- La méthode `withEncryptionMaterials()` du générateur de clients de chiffrement a été renommée `withEncryptionMaterialsProvider ()`. Il s'agit simplement d'un changement cosmétique qui reflète plus précisément le type d'argument. Vous devez utiliser la nouvelle méthode lorsque vous configurez votre client de service.

Note

Lorsque vous déchiffrez avec AES-GCM, lisez l'objet dans son intégralité avant de commencer à utiliser les données déchiffrées. Cela permet de vérifier que l'objet n'a pas été modifié depuis qu'il a été chiffré.

Utiliser des fournisseurs de matériel de chiffrement

Vous pouvez continuer à utiliser les mêmes fournisseurs de matériel de chiffrement et les mêmes objets de matériel de chiffrement que ceux que vous utilisez déjà avec le client de chiffrement V1. Ces classes sont chargées de fournir les clés que le client de chiffrement utilise pour sécuriser vos données. Ils peuvent être utilisés de manière interchangeable avec le client de chiffrement V2 et V1.

Configuration du client de chiffrement V2

Le client de chiffrement V2 est configuré avec un `CryptoConfigurationV2` objet. Cet objet peut être construit en appelant son constructeur par défaut, puis en modifiant ses propriétés selon les besoins à partir des valeurs par défaut.

Les valeurs par défaut pour `CryptoConfigurationV2` sont les suivantes :

- `cryptoMode = CryptoMode.StrictAuthenticatedEncryption`
- `storageMode = CryptoStorageMode.ObjectMetadata`
- `secureRandom=` instance de `SecureRandom`
- `rangeGetMode = CryptoRangeGetMode.DISABLED`
- `unsafeUndecryptableObjectPassthrough = false`

Notez que cela n'`EncryptionOnly` est pas pris en charge `cryptoMode` dans le client de chiffrement V2. Le client de chiffrement V2 chiffre toujours le contenu à l'aide d'un chiffrement authentifié et protège les clés de chiffrement de contenu (CEK) à l'aide d'objets V2. `KeyWrap`

L'exemple suivant montre comment spécifier la configuration cryptographique dans la version 1 et comment instancier un objet `CryptoConfigurationV2` à transmettre au générateur de clients de chiffrement V2.

V1

```
CryptoConfiguration cryptoConfiguration = new CryptoConfiguration()  
    .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

V2

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()  
    .withCryptoMode(CryptoMode.StrictAuthenticatedEncryption);
```

Exemples supplémentaires

Les exemples suivants montrent comment traiter des cas d'utilisation spécifiques liés à une migration de la V1 à la V2.

Configurer un client de service pour lire les objets créés par le client de chiffrement V1

Pour lire des objets précédemment écrits à l'aide d'un client de chiffrement V1, définissez la valeur `cryptoMode` sur `AuthenticatedEncryption`. L'extrait de code suivant montre comment créer un objet de configuration avec ce paramètre.

```
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()  
    .withCryptoMode(CryptoMode.AuthenticatedEncryption);
```

Configuration d'un client de service pour obtenir des plages d'octets d'objets

Pour pouvoir accéder à `get` une plage d'octets à partir d'un objet S3 chiffré, activez le nouveau paramètre de configuration `rangeGetMode`. Ce paramètre est désactivé par défaut sur le client de chiffrement V2. Notez que même lorsqu'elle est activée, une plage `get` ne fonctionne que sur les objets chiffrés à l'aide d'algorithmes pris en charge par le `cryptoMode` paramètre du client. Pour plus d'informations, consultez [CryptoRangeGetMode](#) la référence de AWS SDK for Java l'API.

Si vous prévoyez d'utiliser le Amazon S3 TransferManager pour effectuer des téléchargements partitionnés d' Amazon S3 objets chiffrés à l'aide du client de chiffrement V2, vous devez d'abord activer le `rangeGetMode` paramètre sur le client de chiffrement V2.

L'extrait de code suivant montre comment configurer le client V2 pour effectuer une opération à distance. `get`

```
// Allows range gets using AES/CTR, for V2 encrypted objects only  
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()  
    .withRangeGetMode(CryptoRangeGetMode.ALL);  
  
// Allows range gets using AES/CTR and AES/CBC, for V1 and V2 objects  
CryptoConfigurationV2 cryptoConfiguration = new CryptoConfigurationV2()  
    .withCryptoMode(CryptoMode.AuthenticatedEncryption)  
    .withRangeGetMode(CryptoRangeGetMode.ALL);
```

clé OpenPGP pour le AWS SDK for Java

Tous les artefacts Maven accessibles au public pour le AWS SDK for Java sont signés selon le standard OpenPGP. La clé publique dont vous avez besoin pour vérifier la signature d'un artefact est disponible dans la section suivante.

Clé actuelle

Le tableau suivant présente les informations clés d'OpenPGP pour les versions actuelles du SDK pour Java 1x et du SDK pour Java 2.x.

ID de clé	0xAC107B386692DAjouter
Type	RSA
Size	4096/4096
Créé	2016-06-30
Expire	08/10/2024
ID de l'utilisateur	AWSSDK et outils < aws-dr-tools @amazon .com>
Empreinte digitale clé	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 AJOUTER

Pour copier la clé publique OpenPGP suivante pour le SDK pour Java dans le presse-papiers, sélectionnez l'icône « Copier » dans le coin supérieur droit.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xsFNBFd1gAUBEACqbmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYp0viSWsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej
kT2lPffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WP1nxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie
```



```
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+Uk1gjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+ff+Xf0Cl6by0JFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFEMAuzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVB0HKrq+fluorWepQivctzN6Y1N0kx5naTPGGaKWK7G2q
TbcY5SMnkIwFLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNARpWb3dPMThL2xAY+fs60vXdb1Sk0tYJpDWPfGvo0d+VQ+
hV6Xu1GAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDE0062kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SjQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKU0b6b1786WnySIzF2gxqlkkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7plkBf8QRe6biiQRf3KD
0Sn5CbmXpAcHJ1ZHzRRdkXZDNQC6vCjXsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprismHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs
/Hd981FdVghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VWHwQsaW
jMrGj000MFzXGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvvcMXnduLtkBEX7TISMPW+n+0Ta63/z4YFFEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

Historique du document

Cette rubrique décrit les modifications importantes apportées au Guide du AWS SDK for Java développeur au cours de son histoire.

Cette documentation a été élaborée le 21 mai 2024

21 mai 2024, 2024

Supprimez les instructions pour définir `networkaddress.cache.ttl` la propriété de sécurité à l'aide d'une propriété système de ligne de commande Java. Consultez [Comment configurer le JVM TTL](#).

12 janvier 2024

Ajoutez une bannière annonçant la fin du support pour la AWS SDK for Java v1.x.

6 décembre 2023

- Fournissez [la clé OpenPGP actuelle](#).

14 mars 2023

- Mise à jour du guide s'aligner sur les bonnes pratiques IAM. Pour plus d'informations, consultez [Bonnes pratiques de sécurité dans IAM](#).

28 juillet 2022

- Ajout d'une alerte indiquant qu'EC2-Classic sera retiré le 15 août 2022.

22 mars 2018

- Suppression de la gestion des sessions Tomcat, DynamoDB par exemple, car cet outil n'est plus pris en charge.

2 nov 2017

- Ajout d'exemples de Amazon S3 chiffrement pour le client de chiffrement, y compris de nouvelles rubriques : [utilisation du chiffrement côté Amazon S3 client et du chiffrement côté Amazon S3 client avec des clés gérées par AWS KMS et du chiffrement côté client avec des clés principales Amazon S3 du client](#).

14 avril 2017

- Plusieurs mises à jour ont été apportées à la section [Amazon S3 Exemples d'utilisation](#) de AWS SDK for Java cette section, notamment de nouvelles rubriques : [gestion des autorisations Amazon S3 d'accès pour les compartiments et les objets](#) et [configuration d'un Amazon S3 compartiment en tant que site Web](#).

04 avril 2017

- Une nouvelle rubrique, [Enabling Metrics for the](#), AWS SDK for Java décrit comment générer des mesures de performance des applications et des SDK pour le AWS SDK for Java.

03 avril 2017

- De nouveaux CloudWatch exemples ont été ajoutés aux [CloudWatch exemples d'utilisation de la AWS SDK for Java](#) section : [obtention de métriques à partir de métriques CloudWatch](#), [publication de données métriques personnalisées](#), [utilisation d' CloudWatch alarmes](#), [utilisation d'actions d'alarme](#) et [envoi d'événements à CloudWatch CloudWatch](#)

27 mars 2017

- D'autres Amazon EC2 exemples ont été ajoutés aux [Amazon EC2 exemples d'utilisation de la AWS SDK for Java](#) section : [gestion des Amazon EC2 instances](#), [utilisation d'adresses IP élastiques dans Amazon EC2](#), [utilisation de régions et de zones de disponibilité](#), [utilisation de paires de Amazon EC2 clés](#) et [utilisation de groupes de sécurité dans Amazon EC2](#).

21 mars 2017

- [Ajout d'un nouvel ensemble d'exemples IAM aux exemples IAM à l'aide de la AWS SDK for Java](#) section : [Gestion des clés d'accès IAM](#), [Gestion des utilisateurs IAM](#), [Utilisation des alias de compte IAM](#), [Utilisation des politiques IAM](#) et [Utilisation des certificats de serveur IAM](#)

13 mars 2017

- Trois nouveaux sujets ont été ajoutés à la Amazon SQS section : [activation des longs sondages pour les files d'attente de Amazon SQS messages](#), [définition du délai de visibilité dans Amazon SQS](#) et [utilisation de files d'attente de lettres mortes](#) dans. Amazon SQS

26 janvier 2017

- Ajout d'une nouvelle Amazon S3 rubrique, [Utilisation TransferManager pour les Amazon S3 opérations](#), et d'une nouvelle rubrique [Meilleures pratiques pour le AWS développement](#), dont la [AWS SDK for Java](#) rubrique se trouve dans [la AWS SDK for Java section Utilisation](#).

16 janvier 2017

- Ajout d'une nouvelle Amazon S3 rubrique, [Gestion de l'accès aux Amazon S3 compartiments à l'aide de politiques relatives aux compartiments](#), et de deux nouvelles Amazon SQS rubriques, [Utilisation des files d'attente de Amazon SQS messages](#) et [envoi, réception et suppression Amazon SQS](#) de messages.

16 décembre 2016

- De nouveaux exemples de sujets ont été ajoutés pour DynamoDB : [Travailler avec des tables dans DynamoDB](#) et [Travailler avec des éléments dans DynamoDB](#).

26 septembre 2016

- Les rubriques de la section Avancé ont été déplacées vers [Utilisation du AWS SDK for Java](#), car elles sont vraiment essentielles à l'utilisation du SDK.

25 août 2016

- Une nouvelle rubrique, [Création de clients de service](#), a été ajoutée à [Using the AWS SDK for Java](#), qui montre comment utiliser les créateurs de clients pour simplifier la création de Service AWS clients.

La section [Exemples de AWS SDK for Java code](#) a été mise à jour avec de [nouveaux exemples pour S3](#) qui sont soutenus par un [référentiel](#) contenant l'exemple de code complet. GitHub

02 mai 2016

- Une nouvelle rubrique, [Programmation asynchrone](#), a été ajoutée à la section [Utilisation de la AWS SDK for Java](#) section. Elle décrit comment travailler avec des méthodes clientes asynchrones qui renvoient des Future objets ou qui prennent un AsyncHandler

26 avril 2016

- La rubrique Exigences des certificats SSL a été supprimée, car elle n'est plus pertinente. La prise en charge des certificats signés SHA-1 est obsolète depuis 2015 et le site qui héberge les scripts de test a été supprimé.

14 mars 2016

- Ajout d'une nouvelle rubrique à la Amazon SWF section : [Tâches Lambda](#), qui décrit comment implémenter un Amazon SWF flux de travail qui appelle des Lambda fonctions en tant que tâches au lieu d'utiliser des activités traditionnelles Amazon SWF .

04 mars 2016

- La section [Amazon SWF Exemples d'utilisation de AWS SDK for Java](#) cette section a été mise à jour avec un nouveau contenu :
 - Amazon SWF Notions de [base](#) : fournit des informations de base sur la manière d'inclure le SWF dans vos projets.
 - [Création d'une Amazon SWF application simple](#) - Un nouveau didacticiel qui fournit step-by-step des conseils aux développeurs Java novices Amazon SWF.
 - [Arrêter les travailleurs d'activité et de flux de travail avec élégance](#) - Décrit comment vous pouvez fermer gracieusement les classes de Amazon SWF travailleurs à l'aide des classes de simultanéité de Java.

23 février 2016

- La source du guide du AWS SDK for Java développeur a été déplacée vers [aws-java-developer-guide](#).

28 décembre 2015

- [the section called “Définissez le TTL de la JVM pour les recherches de noms DNS”](#) a été transféré de la version avancée à [l'utilisation du AWS SDK for Java](#), et a été réécrit pour des raisons de clarté.

La section [Utilisation du kit SDK avec Apache Maven](#) a été mise à jour avec des informations sur la manière d'inclure la nomenclature (BOM) du kit SDK dans votre projet.

04 août 2015

- Les exigences relatives aux certificats SSL constituent une nouvelle rubrique de la section [Getting Started](#) qui décrit AWS« le passage à des certificats signés SHA256 pour les connexions SSL » et comment corriger les environnements Java des versions 1.6 et antérieures afin qu'ils utilisent ces certificats, qui sont requis pour y AWS accéder après le 30 septembre 2015.



Note

Java 1.7 (ou version suivante) est déjà capable d'utiliser des certificats signés en SHA256.

14 mai 2014

- Le matériel d'[introduction](#) et de [démarrage](#) a été largement révisé pour soutenir la nouvelle structure du guide et comprend désormais des conseils sur la manière de [configurer les AWS informations d'identification et la région pour le développement](#).

La discussion sur les [exemples de code](#) a été déplacée dans sa propre rubrique au sein de la section [Documentation et ressources supplémentaires](#).

Les informations sur la façon d'[afficher l'historique des révisions du kit SDK](#) ont été déplacées dans l'introduction.

9 mai 2014

- La structure globale de la AWS SDK for Java documentation a été simplifiée, et les rubriques « Mise [en route](#) » et « [Documentation et ressources supplémentaires](#) » ont été mises à jour.

De nouvelles rubriques ont été ajoutées :

- [Travailler avec les AWS informations d'identification](#) : décrit les différentes manières de spécifier les informations d'identification à utiliser avec AWS SDK for Java.
- [Utilisation des rôles IAM pour accorder l'accès aux AWS ressources sur Amazon EC2 : fournit des informations sur](#) la manière de spécifier en toute sécurité les informations d'identification pour les applications exécutées sur des instances EC2.

9 septembre 2013

- Cette rubrique, Historique des documents, suit les modifications apportées au Guide du AWS SDK for Java développeur. Elle résume l'historique des notes de mise à jour.