

Guide du développeur pour la version 2.x

AWS SDK for Java 2.x



AWS SDK for Java 2.x: Guide du développeur pour la version 2.x

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d'Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n'est pas d'Amazon, d'une manière susceptible de créer une confusion parmi les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Guide du développeur - AWS SDK for Java 2.x	1
Commencez avec le SDK	1
Développez des applications mobiles	1
Maintenance et prise en charge des versions majeures du SDK	1
Ressources supplémentaires	2
Contribuez au SDK	2
Didacticiel de premiers pas	3
Étape 1 : Configuration de ce didacticiel	3
Étape 2 : Création du projet	3
Étape 3 : Écrire le code	8
Étape 4 : créer et exécuter l'application	12
Réussite	13
Nettoyage	13
Étapes suivantes	14
Configuration	15
Présentation de la configuration	15
Configurer l'authentification	16
Configuration de l'accès par authentification unique pour le SDK	16
Connectez-vous à l'aide du AWS CLI	17
Installation de Java et d'un outil de compilation	18
Options d'authentification supplémentaires	18
Configuration d'un projet Apache Maven	18
Prérequis	19
Création d'un projet Maven	19
Configuration du compilateur Java pour Maven	20
Déclaration du kit SDK comme dépendance	21
Définition des dépendances pour les modules SDK	22
Génération de votre projet	24
Configurer un projet Gradle	25
Configuration d'un projet GraalVM Native Image	31
Prérequis	32
Créez un projet à l'aide de l'archétype	32
Création d'une image native	33
Utiliser le SDK	34

Travailler avec les clients du service	34
Création d'un client de service	34
Configuration du client par défaut	35
Configuration des clients de service	35
Faites des demandes	36
Gérer les réponses	36
Fermer le client de service	37
Gérer les exceptions	38
Utilisez des serveurs	39
Clients HTTP	39
Nouvelle tentative	40
Délais	40
Intercepteurs d'exécution	41
Informations supplémentaires	41
Fournir des informations d'identification temporaires au SDK	41
Configuration de l'accès aux informations d'identification temporaires	42
Chaîne de fournisseurs d'informations d'identification par défaut	44
Utiliser un fournisseur d'informations d'identification ou une chaîne de fournisseurs spécifique	45
Profils d'utilisation	46
Charger des informations d'identification temporaires à partir d'un processus externe	49
Fournir des informations d'identification temporaires dans le code	53
Lisez les informations d'identification du rôle IAM sur Amazon EC2	56
Utiliser Régions AWS	58
Configurez explicitement un Région AWS	58
Déterminer la région à partir de l'environnement	59
Vérifiez la disponibilité du service	60
Choisissez un point de terminaison spécifique	61
Réduisez le temps de démarrage du SDK pour AWS Lambda	61
Utiliser un client AWS HTTP CRT	61
Supprimer les dépendances du client HTTP inutilisées	62
Configuration des clients de service pour des recherches de raccourcis	63
Initialisez le client SDK en dehors du gestionnaire de fonctions Lambda	64
Minimiser l'injection de dépendance	64
Utilisez un archétype de ciblage Maven AWS Lambda	65
Lambda SnapStart	65

Modifications de la version 2.x qui affectent le temps de démarrage	65
Ressources supplémentaires	65
Clients HTTP	66
Clients disponibles	66
Recommandations des clients	67
Défauts intelligents	70
Configuration du client HTTP basé sur Apache	72
Configuration du client HTTP basé sur une connexion URL	78
Configuration du client HTTP basé sur Netty	84
Configuration de clients AWS HTTP basés sur CRT	90
Configuration des proxys HTTP	101
Gestion des exceptions	106
Pourquoi des exceptions non contrôlées ?	107
AwsServiceException (et sous-classes)	107
SdkClientException	108
Exceptions et comportement en cas de nouvelle tentative	108
Journalisation	109
Fichier de configuration Log4j 2	109
Ajouter une dépendance à la journalisation	109
Erreurs et avertissements spécifiques au SDK	110
Enregistrement du résumé des demandes/réponses	111
Journalisation du SDK au niveau du débogage	112
Activer l'enregistrement des câbles	114
Définissez le TTL de la JVM pour les recherches de noms DNS	120
Comment configurer le JVM TTL	120
Bonnes pratiques	121
Réutilisation d'un client SDK	121
Fermer les flux d'entrée	121
Régler les configurations HTTP	122
Utiliser OpenSSL pour Netty	122
Configurer les délais d'expiration de l'API	122
Utiliser les métriques	124
Résolution des problèmes	124
Réinitialisation de connexion	124
Délai de connexion	125
Le délai de lecture a expiré	126

Expiration du pool de connexions	126
Erreurs de chemin de classe	130
Erreurs de signature	131
Arrêt du pool de connexions	133
Utiliser les fonctionnalités du SDK	135
Caractéristiques générales	135
Fonctionnalités spécifiques au service	135
Travailler avec des résultats paginés	136
Pagination synchrone	136
Pagination asynchrone	139
Sondage sur l'état des ressources	145
Prérequis	145
Utiliser des serveurs	146
Configuration des serveurs	147
Exemples de code	147
Utiliser la programmation asynchrone	148
Opérations autres que le streaming	148
Opérations de streaming	151
Opérations avancées	155
Travaillez avec HTTP/2	156
Utiliser les métriques du SDK	157
Prérequis	157
Comment activer la collecte de métriques	158
Personnaliser l'éditeur de métriques	159
Quand les statistiques sont-elles disponibles ?	161
Quelles sont les informations collectées ?	161
Comment puis-je utiliser ces informations ?	162
Mesures relatives aux clients du service	162
Travaillez avec Services AWS	168
CloudWatch	168
Obtenez des statistiques auprès de CloudWatch	169
Publiez des données métriques personnalisées sur CloudWatch	171
Travailler avec des CloudWatch alarmes	173
Utiliser Amazon CloudWatch Events	177
AWS services de base de données	181
Amazon DynamoDB	181

Amazon RDS	182
Amazon Redshift	182
Amazon Aurora sans serveur v1	183
Amazon DocumentDB	183
DynamoDB	183
Travaillez avec des tables dans DynamoDB	184
Travaillez avec des objets dans DynamoDB	194
Associer des objets à des éléments DynamoDB	201
Amazon EC2	321
Gérer les Amazon EC2 instances	321
Zones d'utilisation Régions AWS et de disponibilité	328
Travaillez avec des groupes de sécurité dans Amazon EC2	331
Utiliser les métadonnées d'instance Amazon EC2	336
IAM	342
Gérer les clés IAM d'accès	343
Gérer les IAM utilisateurs	349
Création de politiques IAM	353
Travailler avec des IAM politiques	361
Travailler avec des certificats IAM de serveur	368
Kinesis	372
S'abonner à Amazon Kinesis Data Streams	373
Lambda	384
Appel d'une fonction Lambda	384
Liste des fonctions Lambda	385
Suppression d'une fonction Lambda	386
Amazon S3	387
Utiliser des points d'accès ou des points d'accès multirégionaux	388
Opérations de compartiment	389
Opérations sur les objets	396
URL pré-signées	406
Accès interrégional	415
Totaux de contrôle	416
Utilisez un client S3 performant	418
Transférer des fichiers et des répertoires	421
Amazon SNS	429
Créer une rubrique	429

Listez vos Amazon SNS sujets	430
Abonner un point de terminaison à une rubrique	431
Publier un message dans une rubrique	432
Désabonner un point de terminaison à une rubrique	433
Supprimer une rubrique	434
Amazon SQS	435
Opérations sur les files	435
Opérations relatives aux messages	439
Amazon Transcribe	442
Configurez le microphone	443
Création d'un éditeur	443
Créez le client et lancez le stream	446
En savoir plus	442
Exemples de code	448
Actions et scénarios	448
API Gateway	450
Application Auto Scaling	454
Contrôleur de restauration d'applications	463
Aurora	466
Auto Scaling	501
Amazon Bedrock	563
Amazon Bedrock Runtime	569
CloudFront	649
CloudWatch	669
CloudWatch Évènements	720
CloudWatch Journaux	726
Amazon Cognito Identity	737
Fournisseur d'identité Amazon Cognito	744
Amazon Comprehend	772
DynamoDB	783
Amazon EC2	862
Amazon ECS	934
Elastic Load Balancing - Version 2	948
MediaStore	993
OpenSearch Service	1009
EventBridge	1017

Forecast	1050
AWS Glue	1063
HealthImaging	1087
IAM	1116
AWS IoT	1201
AWS IoT data	1228
Amazon Keyspaces	1231
Kinesis	1257
AWS KMS	1270
Lambda	1307
MediaConvert	1332
Migration Hub	1355
Amazon Personalize	1368
Amazon Personalize Events	1399
Amazon Personalize Runtime	1402
Amazon Pinpoint	1407
API de messages SMS et vocaux Amazon Pinpoint	1452
Amazon Polly	1456
Amazon RDS	1462
Amazon Redshift	1503
Amazon Rekognition	1529
Enregistrement de domaine Route 53	1597
Amazon S3	1620
S3 Glacier	1762
SageMaker	1778
Secrets Manager	1807
Amazon SES	1809
API Amazon SES v2	1822
Amazon SNS	1840
Amazon SQS	1889
Step Functions	1909
AWS STS	1933
AWS Support	1937
Systems Manager	1960
Amazon Textract	1990
Amazon Transcribe	2001

Exemples de services croisés	2018
Créer une application pour soumettre des données à une table DynamoDB	2019
Création d'un chatbot Amazon Lex	2019
Création d'une application Amazon SNS	2019
Création d'une application de messagerie	2020
Création d'une application sans serveur pour gérer des photos	2020
Créer une application web pour suivre les données DynamoDB	2021
Créer une application web pour suivre les données Amazon Redshift	2021
Créer un outil de suivi des éléments de travail sans serveur Aurora	2022
Créer une application pour analyser les commentaires des clients	2022
Détection de l'EPI dans des images	2023
Détection des objets dans des images	2023
Détection des personnes et des objets dans une vidéo	2024
Surveiller les performances de DynamoDB	2024
Publier des messages dans des files d'attente	2025
Utiliser API Gateway pour invoquer une fonction Lambda	2025
Utiliser les fonctions Step Functions pour appeler des fonctions Lambda	2026
Utiliser des événements planifiés pour invoquer une fonction Lambda	2026
Sécurité	2028
Protection des données	2028
protocole TLS (Transport Layer Security)	2030
Vérifiez les versions TLS	2030
Appliquer les versions TLS	2031
Migrer vers TLS 1.2	2031
Gestion de l'identité et des accès	2031
Public ciblé	2032
Authentification par des identités	2032
Gestion des accès à l'aide de politiques	2036
Comment Services AWS travailler avec IAM	2039
Résolution des problèmes AWS d'identité et d'accès	2039
Validation de la conformité	2041
Résilience	2043
Sécurité de l'infrastructure	2044
Migrer vers la version 2	2045
Nouveautés de la version 2	2045
tep-by-step Instructions S	2046

Présentation des étapes	2046
Exemple de migration	2048
Nom du package vers les mappages ArtifactID	2059
Convention de dénomination	2059
Exceptions	2060
Quelles sont les différences entre 1.x et 2.x	2065
Modification du nom du package	2065
Ajouter la version 2.x à votre projet	2066
POJO immuables	2066
Méthodes Setter et Getter	2067
Noms des classes de modèles	2068
Bibliothèques et utilitaires	2068
Modifications du client	2070
Modifications apportées au fournisseur d'identifiants	2115
Changements de région	2123
Modifications des opérations, des demandes et des réponses	2125
Modifications des exceptions	2126
Changements de sérialisation	2127
Modifications spécifiques au service	2128
Modifications du fichier de profil	2134
Configuration externe	2135
Programmes d'attente	2138
Gestionnaire de transfert S3	2142
Utilitaire de métadonnées EC2	2149
CloudFront présignant	2156
Analyse d'URI S3	2160
API IAM Policy Builder	2163
Utilisation du kit SDK for Java 1.x side-by-side	2169
Clé OpenPGP	2171
Clé actuelle	2171
Historique du document	2173
.....	mmclxxxii

Guide du développeur - AWS SDK for Java 2.x

Le kit AWS SDK for Java fournit une API Java pour Services AWS. À l'aide du SDK, vous pouvez créer des applications Java qui fonctionnent avec Amazon S3, Amazon EC2, DynamoDB, et plus encore.

Le AWS SDK for Java 2.x est une réécriture majeure de la base de code de la version 1.x. Il repose sur Java 8+ et ajoute plusieurs fonctionnalités fréquemment demandées. Il s'agit notamment de la prise en charge des E/S non bloquantes et de la possibilité de connecter une implémentation HTTP différente lors de l'exécution.

Nous ajoutons régulièrement la prise en charge de nouveaux services au kit AWS SDK for Java. Pour obtenir une liste des modifications et fonctions d'une version particulière, consultez le [journal des modifications](#).

Commencez avec le SDK

Si vous êtes prêt à vous familiariser avec le SDK, suivez le [Didacticiel de premiers pas](#) didacticiel.

Pour configurer votre environnement de développement, consultez [Configuration](#).

Si vous utilisez actuellement la version 1.x du SDK for Java, consultez [Migrer vers la version 2](#) pour obtenir des conseils spécifiques.

Pour plus d'informations sur la manière de faire des demandes à Amazon S3, DynamoDB, Amazon EC2 et sur d'autres sujets Services AWS, voir [Utiliser le SDK for Java](#) et [travailler avec Services AWS](#).

Développez des applications mobiles

Si vous êtes un développeur d'applications mobiles, Amazon Web Services fournit le [AWS Amplify](#) framework.

Maintenance et prise en charge des versions majeures du SDK

Pour plus d'informations sur la maintenance et le support des versions majeures du SDK et de leurs dépendances sous-jacentes, consultez les rubriques suivantes du Guide de [référence AWS des SDK et des outils](#) :

- [AWSPolitique de maintenance des SDK et des outils](#)
- [AWSMatrice de support des versions des SDK et des outils](#)

Ressources supplémentaires

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

- [AWS SDK for JavaRéférence de l'API 2.x](#)
- [Blog des développeurs Java](#)
- [Rubrique de développement Java dans AWS re:Post](#)
- [Source du SDK activée](#) GitHub
- [AWSBibliothèque d'exemples de code SDK](#)
- [@awsforjava](#) (Twitter)

Contribuez au SDK

Les développeurs peuvent également contribuer à travers leurs commentaires sur les canaux suivants :

- Soumettez des problèmes sur GitHub :
 - [Soumettre les problèmes liés à la documentation du guide du](#)
 - [Soumettre les problèmes du kit SDK](#)
- [Participez à une discussion informelle sur le SDK sur le canal AWS SDK for Java Gitter 2.x](#)

Commencez avec le AWS SDK for Java 2.x

AWS SDK for Java 2.x fournit des API Java pour Amazon Web Services (AWS). À l'aide du SDK, vous pouvez créer des applications Java qui fonctionnent avec Amazon S3, Amazon EC2, DynamoDB, et plus encore.

Ce didacticiel explique comment utiliser [Apache Maven](#) pour définir les dépendances du SDK pour Java 2.x, puis écrire le code permettant de se connecter Amazon S3 pour télécharger un fichier.

Pour terminer ce didacticiel, procédez comme suit :

- [Étape 1 : Configuration de ce didacticiel](#)
- [Étape 2 : Création du projet](#)
- [Étape 3 : Écrire le code](#)
- [Étape 4 : créer et exécuter l'application](#)

Étape 1 : Configuration de ce didacticiel

Avant de commencer ce didacticiel, vous devez disposer des éléments suivants :

- Autorisation d'accès Amazon S3
- Un environnement de développement Java configuré pour accéder à Services AWS l'aide de l'authentification unique à AWS IAM Identity Center

Suivez les instructions ci-dessous [???](#) pour vous préparer à ce didacticiel. Une fois que vous avez [configuré votre environnement de développement avec un accès par authentification unique](#) pour le SDK Java et que vous disposez d'une [session de portail d' AWS accès active](#), passez à l'étape 2 de ce didacticiel.

Étape 2 : Création du projet

Pour créer le projet de ce didacticiel, vous devez exécuter une commande Maven qui vous invite à saisir des informations sur la configuration du projet. Une fois toutes les entrées saisies et confirmées, Maven termine la création du projet en créant un `pom.xml` et crée des fichiers Java stub.

1. Ouvrez un terminal ou une fenêtre d'invite de commande et naviguez jusqu'au répertoire de votre choix, par exemple, votre Home dossier Desktop ou.
2. Entrez la commande suivante sur le terminal et appuyez sur `Enter`.

```
mvn archetype:generate \
  -DarchetypeGroupId=software.amazon.awssdk \
  -DarchetypeArtifactId=archetype-app-quickstart \
  -DarchetypeVersion=2.20.43
```

3. Entrez la valeur répertoriée dans la deuxième colonne pour chaque invite.

Invite	Valeur à saisir
Define value for property 'service':	s3
Define value for property 'httpClient' :	apache-client
Define value for property 'nativeImage' :	false
Define value for property 'credentialProvider'	identity-center
Define value for property 'groupId':	org.example
Define value for property 'artifactId':	getstarted
Define value for property 'version' 1.0-SNAPSHOT:	<Enter>
Define value for property 'package' org.example:	<Enter>

4. Une fois la dernière valeur saisie, Maven répertorie les choix que vous avez effectués. Confirmez en saisissant `Y` ou `n` à nouveau des valeurs en saisissant `N`.

Maven crée le dossier de projet nommé `getstarted` en fonction de la `artifactId` valeur que vous avez saisie. Dans le `getstarted` dossier, trouvez un `README.md` fichier que vous pouvez consulter, un `pom.xml` fichier et un `src` répertoire.

Maven crée l'arborescence de répertoires suivante.

```
getstarted
### README.md
### pom.xml
### src
  ### main
  #   ### java
  #   #   ### org
  #   #   ### example
  #   #   ### App.java
  #   #   ### DependencyFactory.java
  #   #   ### Handler.java
  #   ### resources
  #   ### simplelogger.properties
  ### test
  ### java
  ### org
  ### example
  ### HandlerTest.java
```

10 directories, 7 files

Ce qui suit montre le contenu du fichier de `pom.xml` projet.

pom.xml

La `dependencyManagement` section contient une dépendance par rapport à Amazon S3 AWS SDK for Java 2.x et possède une dépendance par rapport à Amazon S3. `dependencies` Le projet utilise Java 1.8 en raison de la 1.8 valeur des `maven.compiler.target` propriétés `maven.compiler.source` et.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```



```

<groupId>org.example</groupId>
<artifactId>getstarted</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
  <maven.shade.plugin.version>3.2.1</maven.shade.plugin.version>
  <maven.compiler.plugin.version>3.6.1</maven.compiler.plugin.version>
  <exec-maven-plugin.version>1.6.0</exec-maven-plugin.version>
  <aws.java.sdk.version>2.20.43</aws.java.sdk.version> <----- SDK version
picked up from archetype version.
  <slf4j.version>1.7.28</slf4j.version>
  <junit5.version>5.8.1</junit5.version>
</properties>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>${aws.java.sdk.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId> <----- S3 dependency
    <exclusions>
      <exclusion>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>netty-nio-client</artifactId>
      </exclusion>
      <exclusion>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>apache-client</artifactId>
      </exclusion>
    </exclusions>
  </dependency>

```

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>sso</artifactId> <----- Required for identity center
authentication.
</dependency>

<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>ssooidc</artifactId> <----- Required for identity center
authentication.
</dependency>

<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>apache-client</artifactId> <----- HTTP client specified.
  <exclusions>
    <exclusion>
      <groupId>commons-logging</groupId>
      <artifactId>commons-logging</artifactId>
    </exclusion>
  </exclusions>
</dependency>

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>${slf4j.version}</version>
</dependency>

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>${slf4j.version}</version>
</dependency>

<!-- Needed to adapt Apache Commons Logging used by Apache HTTP Client to Slf4j
to avoid
ClassNotFoundException: org.apache.commons.logging.impl.LogFactoryImpl during
runtime -->
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jcl-over-slf4j</artifactId>
  <version>${slf4j.version}</version>
```

```
</dependency>

<!-- Test Dependencies -->
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter</artifactId>
  <version>${junit5.version}</version>
  <scope>test</scope>
</dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>${maven.compiler.plugin.version}</version>
    </plugin>
  </plugins>
</build>

</project>
```

Étape 3 : Écrire le code

Le code suivant montre la App classe créée par Maven. La main méthode est le point d'entrée dans l'application, qui crée une instance de la Handler classe puis appelle sa sendRequest méthode.

classe **App**

```
package org.example;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class App {
  private static final Logger logger = LoggerFactory.getLogger(App.class);

  public static void main(String... args) {
    logger.info("Application starts");

    Handler handler = new Handler();
    handler.sendRequest();
  }
}
```

```
        logger.info("Application ends");
    }
}
```

La `DependencyFactory` classe créée par Maven contient la méthode `s3Client` factory qui construit et renvoie une [S3Client](#) instance. L'`S3Client` instance utilise une instance du client HTTP basé sur Apache. Cela est dû au fait que vous avez spécifié le client HTTP à utiliser `apache-client` lorsque Maven vous a demandé.

Cela `DependencyFactory` est indiqué dans le code suivant.

classe `DependencyFactory`

```
package org.example;

import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;

/**
 * The module containing all dependencies required by the {@link Handler}.
 */
public class DependencyFactory {

    private DependencyFactory() {}

    /**
     * @return an instance of S3Client
     */
    public static S3Client s3Client() {
        return S3Client.builder()
            .httpClientBuilder(ApacheHttpClient.builder())
            .build();
    }
}
```

La `Handler` classe contient la logique principale de votre programme. Lorsqu'une instance de `Handler` est créée dans la `App` classe, elle `DependencyFactory` fournit le client de `S3Client` service. Votre code utilise l'`S3Client` instance pour appeler le service Amazon S3.

Maven génère la `Handler` classe suivante avec un `TODO` commentaire. L'étape suivante du didacticiel remplace le code `TODO` par.

Handler classe, générée par Maven

```
package org.example;

import software.amazon.awssdk.services.s3.S3Client;

public class Handler {
    private final S3Client s3Client;

    public Handler() {
        s3Client = DependencyFactory.s3Client();
    }

    public void sendRequest() {
        // TODO: invoking the api calls using s3Client.
    }
}
```

Pour compléter la logique, remplacez l'intégralité du contenu de la `Handler` classe par le code suivant. La `sendRequest` méthode est renseignée et les importations nécessaires sont ajoutées.

Handler classe, implémentée

Le code crée d'abord un nouveau compartiment S3 avec la dernière partie du nom généré `System.currentTimeMillis()` afin de rendre le nom du compartiment unique.

Après avoir créé le compartiment dans la `createBucket()` méthode, le programme télécharge un objet à l'aide de la [putObject](#) méthode de `S3Client`. Le contenu de l'objet est une chaîne simple créée avec la `RequestBody.fromString` méthode.

Enfin, le programme supprime l'objet suivi du bucket dans la `cleanup` méthode.

```
package org.example;

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;

public class Handler {
    private final S3Client s3Client;

    public Handler() {
        s3Client = DependencyFactory.s3Client();
    }

    public void sendRequest() {
        String bucket = "bucket" + System.currentTimeMillis();
        String key = "key";

        createBucket(s3Client, bucket);

        System.out.println("Uploading object...");

        s3Client.putObject(PutObjectRequest.builder().bucket(bucket).key(key)
            .build(),
            RequestBody.fromString("Testing with the {sdk-java}"));

        System.out.println("Upload complete");
        System.out.printf("%n");

        cleanUp(s3Client, bucket, key);

        System.out.println("Closing the connection to {S3}");
        s3Client.close();
        System.out.println("Connection closed");
        System.out.println("Exiting...");
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            s3Client.createBucket(CreateBucketRequest
                .builder()
                .bucket(bucketName)
                .build());
            System.out.println("Creating bucket: " + bucketName);
            s3Client.waiter().waitUntilBucketExists(HeadBucketRequest.builder()
                .bucket(bucketName)
                .build());
            System.out.println(bucketName + " is ready.");
        }
    }
}
```

```
        System.out.printf("%n");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void cleanUp(S3Client s3Client, String bucketName, String keyName) {
    System.out.println("Cleaning up...");
    try {
        System.out.println("Deleting object: " + keyName);
        DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder().bucket(bucketName).key(keyName).build();
        s3Client.deleteObject(deleteObjectRequest);
        System.out.println(keyName + " has been deleted.");
        System.out.println("Deleting bucket: " + bucketName);
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder().bucket(bucketName).build();
        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println(bucketName + " has been deleted.");
        System.out.printf("%n");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Cleanup complete");
    System.out.printf("%n");
}
}
```

Étape 4 : créer et exécuter l'application

Une fois que le projet est créé et contient la `Handler` classe complète, créez et exécutez l'application.

1. Assurez-vous que vous disposez d'une session IAM Identity Center active. Pour ce faire, exécutez la AWS Command Line Interface commande `aws sts get-caller-identity` et vérifiez la réponse. Si vous n'avez pas de session active, consultez [cette section](#) pour obtenir des instructions.
2. Ouvrez un terminal ou une fenêtre d'invite de commande et accédez au répertoire de votre projet `getstarted`.

3. Utilisez la commande suivante pour créer votre projet :

```
mvn clean package
```

4. Utilisez la commande suivante pour exécuter l'application.

```
mvn exec:java -Dexec.mainClass="org.example.App"
```

Pour afficher le nouveau bucket et le nouvel objet créés par le programme, effectuez les opérations suivantes.

1. Dans `Handler.java`, commentez la ligne `cleanup(s3Client, bucket, key)` dans la `sendRequest` méthode et enregistrez le fichier.
2. Reconstituez le projet en exécutant `mvn clean package`.
3. Exécutez à nouveau `mvn exec:java -Dexec.mainClass="org.example.App"` pour télécharger l'objet texte une fois de plus.
4. Connectez-vous à [la console S3](#) pour afficher le nouvel objet dans le compartiment nouvellement créé.

Après avoir consulté le fichier, supprimez l'objet, puis supprimez le compartiment.

Réussite

Si votre projet Maven a été créé et s'est exécuté sans erreur, alors félicitations ! Vous avez créé avec succès votre première application Java à l'aide du SDK pour Java 2.x.

Nettoyage

Pour nettoyer les ressources que vous avez créées au cours de ce didacticiel, procédez comme suit :

- Si ce n'est pas déjà fait, dans [la console S3](#), supprimez tous les objets et les compartiments créés lors de l'exécution de l'application.
- Supprimez le dossier du projet (`getstarted`).

Étapes suivantes

Maintenant que vous connaissez les notions de base, vous pouvez en apprendre davantage sur les points suivants :

- [Travailler avec Amazon S3](#)
- [Travailler avec d'autres Amazon Web Services](#) [services de base de données DynamoDB](#)[Amazon EC2, tels que, et divers](#)
- [Utiliser le SDK](#)
- [Sécurité pour AWS SDK for Java](#)

Configurer le AWS SDK for Java 2.x

Cette section fournit des informations sur la façon de configurer votre environnement de développement et les projets pour utiliser le AWS SDK for Java 2.x.

Présentation de la configuration

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

- Le SDK Java doit avoir accès aux informations d'identification pour [authentifier les demandes](#) en votre nom.
- 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 PowerUserAccess AWS gérée sont suffisantes pour la plupart des besoins de développement.
- Un environnement de développement comprenant les éléments suivants :
 - [Fichiers de configuration partagés](#) configurés selon au moins l'une des méthodes suivantes :
 - Le config fichier contient les [paramètres d'authentification unique d'IAM Identity Center](#) afin que le SDK puisse obtenir des informations d'identification. AWS
 - Le `credentials` fichier contient des informations d'identification temporaires.
 - [Installation de Java 8](#) ou version ultérieure.
 - Un [outil d'automatisation de build](#) tel que [Maven](#) ou [Gradle](#).
 - Un éditeur de texte pour travailler avec du code.
 - (Facultatif, mais recommandé) Un IDE (environnement de développement intégré) tel que [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.

- Une session de portail AWS d'accès active lorsque vous êtes prêt à exécuter votre application. Vous utilisez le AWS Command Line Interface pour [lancer le processus de connexion](#) au portail d' AWS accès d'IAM Identity Center.

⚠ Important

Les instructions de cette section de configuration supposent que vous ou votre organisation utilisez IAM Identity Center. Si votre entreprise utilise un fournisseur d'identité externe qui fonctionne indépendamment d'IAM Identity Center, découvrez comment obtenir des informations d'identification temporaires à utiliser par 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.

Configurer l'authentification

La rubrique [Authentification et accès](#) du Guide de référence AWS des SDK et des outils décrit les différentes options d'authentification. Nous vous recommandons de suivre les instructions pour [configurer l'accès à l'IAM Identity Center](#) afin que le SDK puisse acquérir des informations d'identification. Après avoir suivi les instructions, [votre système est configuré](#) pour permettre au SDK d'authentifier les demandes.

Configuration de l'accès par authentification unique pour le SDK

Une fois que vous avez terminé l'étape 2 de la [section Accès par programmation](#) afin que le SDK puisse utiliser l'authentification IAM Identity Center, votre système doit contenir les éléments suivants.

- Le AWS CLI, que vous utilisez pour démarrer une [session de portail d'AWS accès](#) avant d'exécuter votre application.
- `~/.aws/config` Fichier contenant un [profil par défaut](#). Le SDK for Java utilise la configuration du fournisseur de jetons SSO du profil pour obtenir des informations d'identification avant d'envoyer des demandes à AWS. La `sso_role_name` valeur, qui est un rôle IAM connecté à un ensemble d'autorisations IAM Identity Center, doit autoriser l'accès à l'utilisateur dans Services AWS votre application.

Le `config` fichier d'exemple suivant montre un profil par défaut configuré avec la configuration du fournisseur de jetons SSO. Le paramètre `sso_session` du profil fait référence à la section `sso-session` nommée. La `sso-session` section contient les paramètres permettant de lancer une session sur le portail AWS d'accès.

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

Pour plus de détails sur les paramètres utilisés dans la configuration du fournisseur de jetons SSO, consultez la section Configuration du [fournisseur de jetons SSO dans le Guide](#) de référence AWS des SDK et des outils.

Si votre environnement de développement n'est pas configuré pour l'accès par programmation comme indiqué précédemment, suivez [l'étape 2 du Guide de référence des SDK](#).

Connectez-vous à l'aide du AWS CLI

Avant d'exécuter une application qui y accède Services AWS, vous devez disposer d'une session de portail d' AWS accès active afin que le SDK puisse utiliser l'authentification IAM Identity Center pour résoudre les informations d'identification. Exécutez la commande suivante dans le AWS CLI pour vous connecter au portail AWS d'accès.

```
aws sso login
```

Étant donné que vous disposez d'une configuration de profil par défaut, il n'est pas nécessaire d'appeler la commande avec l'option `--profile`. Si la configuration de votre fournisseur de jetons SSO utilise un profil nommé, la commande est `aws sso login --profile named-profile`.

Pour vérifier si vous avez déjà une session active, exécutez la AWS CLI commande suivante.

```
aws sts get-caller-identity
```

La réponse à cette commande doit indiquer le compte IAM Identity Center et l'ensemble d'autorisations configurés dans le fichier partagé `config`.

Note

Si vous disposez déjà d'une session active sur le portail AWS d'accès et que vous l'exécutez `aws sso login`, il ne vous sera pas demandé de fournir des informations d'identification.

Cependant, vous verrez une boîte de dialogue demandant l'autorisation d'accéder `botocore` à vos informations. `botocore` est le fondement du AWS CLI .

Sélectionnez Autoriser pour autoriser l'accès à vos informations pour le AWS CLI et le SDK for Java.

Installation de Java et d'un outil de compilation

Votre environnement de développement a besoin des éléments suivants :

- Java 8 ou version ultérieure. [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 Corretto telles que Red Hat OpenJDK et Adoptium.](#)
- Outil de compilation ou IDE compatible avec Maven Central, tel qu'Apache Maven, Gradle ou IntelliJ.
 - Pour plus d'informations sur l'installation et l'utilisation de Maven, consultez <https://maven.apache.org/>.
 - Pour plus d'informations sur l'installation et l'utilisation de Gradle, consultez <https://gradle.org/>.
 - [Pour plus d'informations sur l'installation et l'utilisation d'IntelliJ IDEA, consultez https://www.jetbrains.com/idea/](https://www.jetbrains.com/idea/).

Options d'authentification supplémentaires

Pour plus d'options d'authentification pour le SDK, telles que l'utilisation de profils et de variables d'environnement, consultez le chapitre sur la [configuration](#) du Guide de référence AWS des SDK et des outils.

Configuration d'un projet Apache Maven

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

Prérequis

Pour utiliser le kit AWS SDK for Java avec Maven, vous avez besoin des éléments suivants :

- Java version 8.0 ou ultérieure. Vous pouvez télécharger la dernière version de Java SE Development Kit depuis <http://www.oracle.com/technetwork/java/javase/downloads/>. Le kit AWS SDK for Java fonctionne également avec [OpenJDK](#) et Amazon Corretto, une distribution d'Open Java Development Kit (OpenJDK). Téléchargez la dernière version d'OpenJDK depuis <https://openjdk.java.net/install/index.html>. Téléchargez la dernière version Amazon Corretto 8 ou Amazon Corretto 11 depuis [la Corretto page](#).
- Apache Maven. Si vous devez installer Maven, accédez à <http://maven.apache.org/> pour le télécharger et l'installer.

Création d'un projet Maven

Pour créer un projet Maven à partir de la ligne de commande, exécutez la commande suivante à partir d'un terminal ou d'une fenêtre d'invite de commande.

```
mvn -B archetype:generate \  
-DarchetypeGroupId=software.amazon.awssdk \  
-DarchetypeArtifactId=archetype-lambda -Dservice=s3 -Dregion=US_WEST_2 \  
-DarchetypeVersion=2.X.X \  
-DgroupId=com.example.myapp \  
-DartifactId=myapp
```

Note

Remplacez `com.example.myapp` par l'espace de noms complet du package de votre application. Remplacez également `myapp` par le nom de votre projet. Cette valeur devient le nom du répertoire de votre projet.

Pour utiliser la dernière version de l'archétype, remplacez la version `2.X.X` par la [dernière](#) version de Maven Central.

Cette commande crée un projet Maven à l'aide de la boîte à outils de modélisation des archétypes. L'archétype génère l'échafaudage d'un projet de gestionnaire de fonctions. AWS Lambda Cet archétype de projet est préconfiguré pour être compilé avec Java SE 8 et inclut une dépendance à la version du SDK pour Java 2.x spécifiée avec. `-DarchetypeVersion`

Pour de plus amples informations sur la création et la configuration de projets Maven, veuillez consulter le document [Maven Getting Started Guide](#).

Configuration du compilateur Java pour Maven

Si vous avez créé votre projet en utilisant l'archétype du AWS Lambda projet tel que décrit précédemment, la configuration du compilateur Java est déjà terminée pour vous.

Pour vérifier que cette configuration est présente, commencez par ouvrir le fichier `pom.xml` à partir du dossier de projet que vous avez créé (par exemple, `myapp`) lors de l'exécution de la commande précédente. Regardez le paramètre de version du compilateur Java pour ce projet Maven aux lignes 11 et 12, ainsi que l'inclusion requise du plug-in du compilateur Maven aux lignes 71 à 75.

```
<project>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>${maven.compiler.plugin.version}</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Si vous créez votre projet avec un archétype différent ou en utilisant une autre méthode, vous devez vous assurer que le plug-in du compilateur Maven fait partie de la génération et que ses propriétés source et cible sont toutes deux définies sur 1.8 dans le fichier `pom.xml`.

L'extrait précédent présente une façon de configurer ces paramètres requis.

Vous pouvez également définir la configuration du compilateur en ligne avec la déclaration du plug-in, comme suit.

```
<project>
  <build>
    <plugins>
```

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
  </configuration>
</plugin>
</plugins>
</build>
</project>
```

Déclaration du kit SDK comme dépendance

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.

Si vous avez créé votre projet à l'aide de l'archétype de projet tel que décrit précédemment, la dernière version du SDK est déjà configurée en tant que dépendance dans votre projet.

L'archétype génère une dépendance à un artefact BOM (nomenclature) pour l'identifiant du groupe `software.amazon.awssdk`. Avec une nomenclature, il n'est pas nécessaire de spécifier la version Maven pour les dépendances d'artefacts individuels qui partagent le même identifiant de groupe.

Si vous avez créé votre projet Maven d'une autre manière, configurez la dernière version du kit SDK pour votre projet en vous assurant que le fichier `pom.xml` contient les éléments suivants.

```
<project>
  <properties>
    <aws.java.sdk.version>2.X.X</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
```



```
</project>
```

Note

Remplacez `2.X.X` dans le `pom.xml` fichier par la [dernière version du](#) AWS SDK for Java 2.x

Définition des dépendances pour les modules SDK

Après avoir configuré le kit SDK, vous pouvez ajouter des dépendances pour un ou plusieurs modules du kit AWS SDK for Java à utiliser dans votre projet.

Bien que vous puissiez spécifier le numéro de version pour chaque composant, cela n'est pas nécessaire car vous avez déjà déclaré la version du SDK dans la `dependencyManagement` section à l'aide de l'artefact de nomenclature. Pour charger une version différente d'un module donné, spécifiez un numéro de version pour sa dépendance.

Si vous avez créé votre projet à l'aide de l'archétype de projet tel que décrit précédemment, votre projet est déjà configuré avec plusieurs dépendances. Il s'agit notamment des dépendances pour les gestionnaires de AWS Lambda fonctions et Amazon S3, comme suit.

```
<project>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
      <exclusions>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>netty-nio-client</artifactId>
        </exclusion>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>apache-client</artifactId>
        </exclusion>
      </exclusions>
    </dependency>

    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>url-connection-client</artifactId>
```

```
</dependency>

<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-lambda-java-core</artifactId>
  <version>${aws.lambda.java.version}</version>
</dependency>
</dependencies>
</project>
```

Note

Dans l'exemple ci-dessus, les dépendances proviennent de groupIds différents. La dépendance `aws-lambda-java-core` vient de `software.amazon.awssdk`, alors que la dépendance `aws-lambda-java-core` vient de `com.amazonaws`. La configuration de gestion des dépendances des nomenclatures affecte les artefacts `software.amazon.awssdk`, une version est donc nécessaire pour l'artefact `aws-lambda-java-core`.

Pour le développement de gestionnaires de fonctions Lambda à l'aide du SDK pour Java 2.x `aws-lambda-java-core`, quelle est la bonne dépendance ? Toutefois, si votre application doit gérer des ressources Lambda, l'utilisation d'opérations telles que `listFunctions`, `deleteFunction`, et `invokeFunctioncreateFunction`, votre application nécessite la dépendance suivante.

```
<groupId>software.amazon.awssdk</groupId>
<artifactId>lambda</artifactId>
```

Note

La dépendance `aws-lambda-java-core` exclut les dépendances transitives `netty-nio-client` et les dépendances `apache-client` transitives. À la place de l'un ou l'autre de ces clients HTTP, l'archétype inclut la dépendance `url-connection-client`, ce qui permet de [réduire la latence de démarrage des AWS Lambda fonctions](#).

Ajoutez à votre projet les modules Service AWS et les fonctionnalités dont vous avez besoin pour votre projet. Les modules (dépendances) gérés par le AWS SDK for Java BOM sont répertoriés dans le référentiel [central Maven](#).

Note

Vous pouvez rechercher le fichier `pom.xml` à partir d'un exemple de code pour déterminer les dépendances dont vous avez besoin pour votre projet. Par exemple, si vous êtes intéressé par les dépendances du service DynamoDB, [consultez cet](#) exemple extrait du référentiel d'exemples de [code AWS sur](#) GitHub (Recherchez le `pom.xml` fichier sous [/javav2/example_code/dynamodb.](#))

Construction de l'intégralité du kit SDK dans votre projet

Pour optimiser votre application, nous vous recommandons vivement d'extraire uniquement les composants dont vous avez besoin au lieu de l'intégralité du kit SDK. Cependant, pour construire l'intégralité du kit AWS SDK for Java dans votre projet, déclarez-le dans votre fichier `pom.xml`, comme suit.

```
<project>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-sdk-java</artifactId>
      <version>2.X.X</version>
    </dependency>
  </dependencies>
</project>
```

Génération de votre projet

Après avoir configuré le fichier `pom.xml`, vous pouvez utiliser Maven pour construire votre projet.

Pour construire votre projet Maven à partir de la ligne de commande, ouvrez une fenêtre de terminal ou d'invite de commande, accédez à votre répertoire de projet (par exemple, `myapp`), entrez ou collez la commande suivante, puis appuyez sur Entrée ou Retour.

```
mvn package
```

Cela crée un fichier `.jar` unique (JAR) dans le répertoire `target` (par exemple, `myapp/target`). Ce fichier JAR contient tous les modules de kit SDK que vous avez spécifiés comme dépendances dans votre fichier `pom.xml`.

Configurer un projet Gradle

Vous pouvez utiliser [Gradle](#) pour configurer et créer des AWS SDK for Java projets.

Les étapes initiales de l'exemple suivant proviennent du [guide de démarrage de Gradle](#) pour la version 8.4. Si vous utilisez une version différente, les résultats peuvent légèrement différer.

Pour créer une application Java avec Gradle (ligne de commande)

1. Créez un répertoire pour héberger votre projet. Dans cet exemple, `demo` c'est le nom du répertoire.
2. Dans le `demo` répertoire, exécutez la `gradle init` commande et fournissez les valeurs surlignées en rouge, comme indiqué dans la sortie de ligne de commande suivante. Pour la procédure pas à pas, nous avons choisi Kotlin comme langage DSL pour les scripts de construction, mais un exemple complet pour Groovy est également présenté à la fin de cette rubrique.

```
> gradle init
Starting a Gradle Daemon (subsequent builds will be faster)

Select type of project to generate:
1: basic
2: application
3: library
4: Gradle plugin
Enter selection (default: basic) [1..4] 2

Select implementation language:
1: C++
2: Groovy
3: Java
4: Kotlin
5: Scala
6: Swift
Enter selection (default: Java) [1..6] 3

Generate multiple subprojects for application? (default: no) [yes, no] no
Select build script DSL:
1: Kotlin
2: Groovy
Enter selection (default: Kotlin) [1..2] <Enter>
```

```
Select test framework:
1: JUnit 4
2: TestNG
3: Spock
4: JUnit Jupiter
Enter selection (default: JUnit Jupiter) [1..4] 4

Project name (default: demo): <Enter>
Source package (default: demo): <Enter>
Enter target version of Java (min. 7) (default: 11): <Enter>
Generate build using new APIs and behavior (some features may change in the next
  minor release)? (default: no) [yes, no] <Enter>

> Task :init
To learn more about Gradle by exploring our Samples at https://docs.gradle.org/8.4/samples/sample\_building\_java\_applications.html

BUILD SUCCESSFUL in 3m 43s
2 actionable tasks: 2 executed
```

3. Une fois la `init` tâche terminée, le `demo` répertoire contient la structure arborescente suivante. Nous examinerons de plus près le fichier de compilation principal `build.gradle.kts` (surligné en rouge) dans la section suivante.

```
### app
#   ### build.gradle.kts
#   ### src
#     ### main
#     #   ### java
#     #   #   ### demo
#     #   #     ### App.java
#     #   ### resources
#     ### test
#     #   ### java
#     #   #   ### demo
#     #   #     ### AppTest.java
#     #   ### resources
### gradle
#   ### wrapper
#     ### gradle-wrapper.jar
#     ### gradle-wrapper.properties
### gradlew
### gradlew.bat
```

```
### settings.gradle.kts
```

Le `build.gradle.kts` fichier contient le contenu échafaudé suivant.

```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java application project to get you
 * started.
 * For more details on building Java & JVM projects, please refer to https://
 * docs.gradle.org/8.4/userguide/building_java_projects.html in the Gradle
 * documentation.
 */

plugins {
    // Apply the application plugin to add support for building a CLI application
    // in Java.
    application
}

repositories {
    // Use Maven Central for resolving dependencies.
    mavenCentral()
}

dependencies {
    // Use JUnit Jupiter for testing.
    testImplementation("org.junit.jupiter:junit-jupiter:5.9.3")

    testRuntimeOnly("org.junit.platform:junit-platform-launcher")

    // This dependency is used by the application.
    implementation("com.google.guava:guava:32.1.1-jre")
}

// Apply a specific Java toolchain to ease working on different environments.
java {
    toolchain {
        languageVersion.set(JavaLanguageVersion.of(11))
    }
}

application {
```

```
// Define the main class for the application.
mainClass.set("demo.App")
}

tasks.named<Test>("test") {
    // Use JUnit Platform for unit tests.
    useJUnitPlatform()
}
```

4. Utilisez le fichier de construction Gradle échafaudé comme base de votre projet. AWS

- a. Pour gérer les dépendances du SDK pour votre projet Gradle, ajoutez la nomenclature Maven (BOM) AWS SDK for Java 2.x pour la dependencies section du fichier. `build.gradle.kts`

```
...
dependencies {
    implementation(platform("software.amazon.awssdk:bom:2.21.1"))
    // With the bom declared, you specify individual SDK dependencies without a
    // version.
    ...
}
...
```

Note

Dans cet exemple de fichier de compilation, remplacez 2.21.1 par la dernière version du SDK pour Java 2.x. Trouvez la dernière version disponible dans le [référentiel central de Maven](#).

- b. Spécifiez les modules du SDK dont votre application a besoin dans la dependencies section. À titre d'exemple, ce qui suit ajoute une dépendance à Amazon Simple Storage Service.

```
...
dependencies {
    implementation(platform("software.amazon.awssdk:bom:2.21.1"))
    implementation("software.amazon.awssdk:s3")
    ...
}
```

...

Gradle résout automatiquement la version correcte des dépendances déclarées en utilisant les informations de la nomenclature.

Les exemples suivants montrent des fichiers de construction Gradle complets dans les DSL Kotlin et Groovy. Le fichier de compilation contient les dépendances pour Amazon S3, l'authentification, la journalisation et les tests. La version source et cible de Java est la version 11.

Kotlin DSL (build.gradle.kts)

```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java application project to get you
 * started.
 * For more details on building Java & JVM projects, please refer to https://
 * docs.gradle.org/8.4/userguide/building_java_projects.html in the Gradle
 * documentation.
 */

plugins {
    // Apply the application plugin to add support for building a CLI application in
    // Java.
    application
}

repositories {
    // Use Maven Central for resolving dependencies.
    mavenCentral()
}

dependencies {
    implementation(platform("software.amazon.awssdk:bom:2.20.56"))
    implementation("software.amazon.awssdk:s3")
    implementation("software.amazon.awssdk:sso")
    implementation("software.amazon.awssdk:ssoidc")
    implementation(platform("org.apache.logging.log4j:log4j-bom:2.20.0"))
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")
    implementation("org.apache.logging.log4j:log4j-1.2-api")
    testImplementation(platform("org.junit:junit-bom:5.10.0"))
    testImplementation("org.junit.jupiter:junit-jupiter")
}
```



```
}

// Apply a specific Java toolchain to ease working on different environments.
java {
    toolchain {
        languageVersion.set(JavaLanguageVersion.of(11))
    }
}

application {
    // Define the main class for the application.
    mainClass.set("demo.App")
}

tasks.named<Test>("test") {
    // Use JUnit Platform for unit tests.
    useJUnitPlatform()
}
```

Groovy DSL (build.gradle)

```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java application project to get you
 * started.
 * For more details on building Java & JVM projects, please refer to https://docs.gradle.org/8.4/userguide/building\_java\_projects.html in the Gradle
 * documentation.
 */

plugins {
    // Apply the application plugin to add support for building a CLI application in
    // Java.
    id 'application'
}

repositories {
    // Use Maven Central for resolving dependencies.
    mavenCentral()
}

dependencies {
```

```
implementation platform('software.amazon.awssdk:bom:2.21.1')
implementation 'software.amazon.awssdk:s3'
implementation 'software.amazon.awssdk:sso'
implementation 'software.amazon.awssdk:ssooidc'
implementation platform('org.apache.logging.log4j:log4j-bom:2.20.0')
implementation 'org.apache.logging.log4j:log4j-slf4j2-impl'
implementation 'org.apache.logging.log4j:log4j-1.2-api'
testImplementation platform('org.junit:junit-bom:5.10.0')
testImplementation 'org.junit.jupiter:junit-jupiter'
}

// Apply a specific Java toolchain to ease working on different environments.
java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(11)
    }
}

application {
    // Define the main class for the application.
    mainClass = 'demo_groovy.App'
}

tasks.named('test') {
    // Use JUnit Platform for unit tests.
    useJUnitPlatform()
}
```

Pour les étapes suivantes, consultez le guide de démarrage sur le site Web de Gradle pour obtenir des instructions sur la [création et l'exécution d'une application Gradle](#).

Configurez un projet GraalVM Native Image pour AWS SDK for Java

Avec les versions 2.16.1 et ultérieures, il AWS SDK for Java prend en out-of-the-box charge les applications GraalVM Native Image. Utilisez l'archétype `archetype-app-quickstart` Maven pour configurer un projet avec un support d'image natif intégré.

Prérequis

- Suivez les étapes décrites dans [Configuration de la version AWS SDK for Java 2.x](#).
- Installez [GraalVM Native Image](#).

Créez un projet à l'aide de l'archétype.

Pour créer un projet Maven avec un support d'image natif intégré, utilisez la commande suivante dans un terminal ou une fenêtre d'invite de commande.

Note

`com.example.mynativeimageapp` Remplacez-le par l'espace de noms complet du package de votre application. Remplacez également `mynativeimageapp` par le nom de votre projet. Cette valeur devient le nom du répertoire de votre projet.

```
mvn archetype:generate \  
  -DarchetypeGroupId=software.amazon.awssdk \  
  -DarchetypeArtifactId=archetype-app-quickstart \  
  -DarchetypeVersion=2.16.1 \  
  -DnativeImage=true \  
  -DhttpClient=apache-client \  
  -Dservice=s3 \  
  -DgroupId=com.example.mynativeimageapp \  
  -DartifactId=mynativeimageapp \  
  -DinteractiveMode=false
```

Cette commande crée un projet Maven configuré avec des dépendances pour le `AWS SDK for Java Amazon S3`, et le `client Apache HttpClient HTTP`. Il inclut également une dépendance pour le [plugin GraalVM Native Image Maven](#), afin que vous puissiez créer des images natives à l'aide de Maven.

Pour inclure les dépendances d'un service différent Amazon Web Services, définissez la valeur du `-Dservice` paramètre sur l'ID d'artefact de ce service. Exemples : `dynamodb`, `comprehend` et `pinpoint`. Pour obtenir la liste complète des identifiants d'artefacts, consultez la liste des dépendances gérées pour [software.amazon.awssdk sur Maven Central](#).

Pour utiliser un client HTTP asynchrone, définissez le `-DhttpClient` paramètre sur `netty-nio-client`. Pour l'utiliser `URLConnectionHttpClient` comme client HTTP synchrone au lieu de `apache-client`, définissez le `-DhttpClient` paramètre sur `url-connection-client`.

Création d'une image native

Une fois le projet créé, exécutez la commande suivante à partir du répertoire de votre projet, par exemple `mynativeimageapp` :

```
mvn package -P native-image
```

Cela crée une application d'imagerie native dans le `target` répertoire, par exemple `target/mynativeimageapp`.

Utilisez le AWS SDK for Java 2.x

Une fois les étapes de [configuration du SDK](#) terminées, vous êtes prêt à adresser des demandes à AWS des services tels qu'Amazon S3, DynamoDB, IAM, Amazon EC2, etc.

Travailler avec les clients du service

Création d'un client de service

Pour envoyer une demande à un Service AWS, vous devez d'abord instancier un client de service pour ce service en utilisant la méthode static `factory.builder()`. La `builder()` méthode renvoie un `builder` objet qui vous permet de personnaliser le client de service. Les méthodes `setter` courantes renvoient l'objet `builder` pour vous permettre de chaîner les appels de méthode pour plus de commodité et pour obtenir un code plus lisible. Après avoir configuré les propriétés souhaitées, appelez la `build()` méthode pour créer le client.

À titre d'exemple, l'extrait de code suivant instancie un `Ec2Client` objet en tant que client de service pour Amazon EC2.

```
Region region = Region.US_WEST_2;
Ec2Client ec2Client = Ec2Client.builder()
    .region(region)
    .build();
```

Note

Les clients de service du kit SDK sont thread-safe. Pour de meilleures performances, traitez-les comme des objets à longue durée de vie. Chaque client possède sa propre ressource de groupe de connexion qui est libérée lorsque le client est nettoyé de la mémoire.

Un objet client de service est immuable. Vous devez donc créer un nouveau client pour chaque service auquel vous envoyez des demandes, ou si vous souhaitez utiliser une configuration différente pour envoyer des demandes au même service.

Il n'est pas nécessaire de spécifier le générateur de clients `Region` dans le service pour tous les AWS services ; toutefois, il est recommandé de définir la région pour les appels d'API que vous effectuez dans vos applications. Voir [AWS la sélection des régions](#) pour plus d'informations.

Configuration du client par défaut

Les générateurs client ont une autre méthode de fabrication nommée `create()`. Cette méthode crée un client de service avec la configuration par défaut. Il utilise 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 `create` échoue. Consultez les [sections Utilisation des informations d'identification](#) et [Sélection de la région](#) pour plus d'informations sur la manière dont le SDK détermine les informations d'identification et la région à utiliser.

Par exemple, l'extrait de code suivant instancie un `DynamoDbClient` objet en tant que client de service pour Amazon DynamoDB :

```
DynamoDbClient dynamoDbClient = DynamoDbClient.create();
```

Configuration des clients de service

Pour personnaliser la configuration d'un client de service, utilisez les paramètres de la méthode `builder()` d'usine. Pour plus de commodité et pour créer un code plus lisible, enchaînez les méthodes pour définir plusieurs options de configuration.

L'exemple suivant `S3Client` montre un système configuré avec plusieurs paramètres personnalisés.

```
ClientOverrideConfiguration clientOverrideConfiguration =
    ClientOverrideConfiguration.builder()
        .apiCallAttemptTimeout(Duration.ofSeconds(1))
        .retryPolicy(RetryPolicy.builder().numRetries(10).build())
        .addMetricPublisher(CloudWatchMetricPublisher.create())
        .build();

Region region = Region.US_WEST_2;
S3Client s3Client = S3Client.builder()
    .region(region)

    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .overrideConfiguration(clientOverrideConfiguration)
    .httpClientBuilder(ApacheHttpClient.builder())

    .proxyConfiguration(proxyConfig.build(ProxyConfiguration.builder()))
    .build();
```

```
.build();
```

Faites des demandes

Utilisez le client de service pour faire des demandes au correspondant Service AWS.

Par exemple, cet extrait de code montre comment créer un `RunInstancesRequest` objet pour créer une nouvelle instance Amazon EC2 :

```
// Create the request by using the fluid setter methods of the request builder.
RunInstancesRequest runInstancesRequest = RunInstancesRequest.builder()
    .imageId(amiId)
    .instanceType(InstanceType.T1_MICRO)
    .maxCount(1)
    .minCount(1)
    .build();

// Use the configured request with the service client.
RunInstancesResponse response = ec2Client.runInstances(runInstancesRequest);
```

Plutôt que de créer une demande et de transmettre l'instance, le SDK fournit des générateurs que vous pouvez utiliser pour créer une demande. Avec un générateur, vous pouvez utiliser une expression Java lambda pour créer la demande « en ligne ».

L'exemple suivant réécrit l'exemple précédent en utilisant la version de la `runInstances` [méthode qui utilise un générateur](#) pour créer la demande.

```
// Create the request by using a lambda expression.
RunInstancesResponse response = ec2.runInstances(r -> r
    .imageId(amiId)
    .instanceType(InstanceType.T1_MICRO)
    .maxCount(1)
    .minCount(1));
```

Gérer les réponses

Le SDK renvoie un objet de réponse pour la plupart des opérations de service. Votre code peut traiter les informations contenues dans l'objet de réponse en fonction de vos besoins.

Par exemple, l'extrait de code suivant affiche le premier identifiant d'instance renvoyé avec l'[RunInstancesResponse](#) objet de la demande précédente.

```
RunInstancesResponse runInstancesResponse =
    ec2Client.runInstances(runInstancesRequest);
System.out.println(runInstancesResponse.instances().get(0).instanceId());
```

Cependant, toutes les opérations ne renvoient pas d'objet de réponse contenant des données spécifiques au service. Dans ces situations, vous pouvez demander l'état de la réponse HTTP pour savoir si l'opération a réussi.

Par exemple, le code de l'extrait suivant vérifie la réponse HTTP pour vérifier si le [DeleteContactList](#) fonctionnement d'Amazon Simple Email Service s'est déroulé correctement.

```
SesV2Client sesv2Client = SesV2Client.create();

DeleteContactListRequest request = DeleteContactListRequest.builder()
    .contactListName("ExampleContactListName")
    .build();

DeleteContactListResponse response = sesv2Client.deleteContactList(request);
if (response.sdkHttpResponse().isSuccessful()) {
    System.out.println("Contact list deleted successfully");
} else {
    System.out.println("Failed to delete contact list. Status code: " +
        response.sdkHttpResponse().statusCode());
}
```

Fermer le client de service

Il est recommandé d'utiliser un client de service pour plusieurs appels de service d'API au cours du cycle de vie d'une application. Toutefois, si vous avez besoin d'un client de service pour une utilisation unique ou si vous n'avez plus besoin du client de service, fermez-le.

Appelez la `close()` méthode lorsque le client de service n'est plus nécessaire pour libérer des ressources.

```
ec2Client.close();
```

Si vous avez besoin d'un client de service pour une utilisation unique, vous pouvez instancier le client de service en tant que ressource dans une instruction `try-with-resources`. Les clients du service implémentent l'[Autoclosable](#) interface, de sorte que le JDK appelle automatiquement la `close()` méthode à la fin de l'instruction.

L'exemple suivant montre comment utiliser un client de service pour un appel ponctuel. Le `StsClient` qui appelle le AWS Security Token Service est fermé après avoir renvoyé l'identifiant du compte.

```
import software.amazon.awssdk.services.sts.StsClient;

String getAccountID() {
    try (StsClient stsClient = StsClient.create()) {
        return stsClient.getCallerIdentity().account();
    }
}
```

Gérer les exceptions

Le SDK utilise des exceptions d'exécution (ou non contrôlées), vous permettant ainsi de contrôler avec précision la gestion des erreurs et de garantir que la gestion des exceptions évolue en fonction de votre application.

Une [SdkServiceException](#), ou l'une de ses sous-classes, est la forme d'exception la plus courante émise par le SDK. Ces exceptions représentent les réponses du AWS service. Vous pouvez également gérer un [SdkClientException](#) problème qui se produit en cas de problème côté client (c'est-à-dire dans votre environnement de développement ou d'application), tel qu'une panne de connexion réseau.

Cet extrait de code montre une façon de gérer les exceptions de service lorsque vous chargez un fichier vers Amazon S3. L'exemple de code détecte à la fois les exceptions du client et du serveur, enregistre les détails et fait exister l'application.

```
Region region = Region.US_WEST_2;
S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

try {

    PutObjectRequest putObjectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();
```

```
s3Client.putObject(putObjectRequest, RequestBody.fromString("SDK for Java test"));

} catch (S3Exception se) {
    System.err.println("Service exception thrown.");
    System.err.println(se.awsErrorDetails().errorMessage());
} catch (SdkClientException ce){
    System.err.println("Client exception thrown.");
    System.err.println(ce.getMessage());
} finally {
    System.exit(1);
}
```

Voir [Gestion des exceptions](#) pour plus d'informations.

Utilisez des serveurs

Le traitement de certaines demandes prend du temps, comme la création d'une nouvelle table DynamoDB ou d'un nouveau Amazon S3 compartiment. Pour vous assurer que la ressource est prête avant que votre code ne continue de s'exécuter, utilisez un serveur.

Par exemple, cet extrait de code crée une nouvelle table (« MyTable ») dans DynamoDB, attend que la table ait un ACTIVE statut, puis imprime la réponse :

```
DynamoDbClient dynamoDbClient = DynamoDbClient.create();
DynamoDbWaiter dynamoDbWaiter = dynamoDbClient.waiter();

WaiterResponse<DescribeTableResponse> waiterResponse =
    dynamoDbWaiter.waitUntilTableExists(r -> r.tableName("myTable"));

waiterResponse.matched().response().ifPresent(System.out::println);
```

Voir [Utilisation de serveurs](#) pour plus d'informations.

Clients HTTP

Vous pouvez modifier la configuration par défaut des clients HTTP dans les applications que vous créez à l'aide du AWS SDK for Java. Pour plus d'informations sur la configuration des clients et des paramètres HTTP, consultez la section [Configuration HTTP](#).

Nouvelle tentative

Vous pouvez modifier les paramètres par défaut pour les nouvelles tentatives chez vos clients de service, notamment le mode de nouvelle tentative et la stratégie de réduction. Pour plus d'informations, reportez-vous à [la `RetryPolicy` classe](#) dans la référence de l' AWS SDK for Java API.

Pour plus d'informations sur les nouvelles tentatives dans les AWS services, consultez la section [Rétentatives d'erreur et ralentissement exponentiel](#) dans. AWS

Délais

Vous pouvez configurer les délais d'expiration pour chacun de vos clients de service à l'aide [`apiCallAttemptTimeout`](#) des paramètres [`apiCallTimeout`](#) et du [`ClientOverrideConfiguration.Builder`](#). Le `apiCallTimeout` paramètre est le délai imparti au client pour terminer l'exécution d'un appel d'API. Le `apiCallAttemptTimeout` paramètre est le temps d'attente avant que chaque requête HTTP (nouvelle tentative) soit terminée avant d'abandonner.

L'exemple suivant définit les deux délais d'expiration pour un client S3.

```
S3Client s3Client = S3Client.builder()
    .overrideConfiguration(b -> b
        .apiCallTimeout(Duration.ofSeconds(105L))
        .apiCallAttemptTimeout(Duration.ofSeconds(25L)))
    build();
```

Vous pouvez également définir des délais d'expiration au niveau de la demande en configurant un [`AwsRequestOverrideConfiguration`](#) et en le fournissant à l'objet de la demande avec la `overrideConfiguration` méthode.

L'exemple suivant utilise les mêmes paramètres de délai d'attente, mais au niveau de la demande pour une `PutObject` opération S3.

```
S3Client basicS3Client = S3Client.create(); // Client with no timeout settings.

AwsRequestOverrideConfiguration overrideConfiguration =
    AwsRequestOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofSeconds(105L))
```

```
.apiCallAttemptTimeout(Duration.ofSeconds(25L))
    .build();

basicS3Client.putObject(b -> b
    .bucket("DOC-EXAMPLE-BUCKET")
    .key("DOC-EXAMPLE_KEY")
    .overrideConfiguration(overrideConfiguration),
    RequestBody.fromString("test"));
```

Intercepteurs d'exécution

Vous pouvez écrire du code qui intercepte l'exécution de vos demandes et réponses d'API à différentes étapes du cycle de vie des demandes/réponses. Cela vous permet de publier des métriques, de modifier une demande en cours de route, de déboguer le traitement des demandes, de consulter les exceptions, etc. Pour plus d'informations, consultez [l'ExecutionInterceptorinterface](#) dans la référence de l' AWS SDK for Java API.

Informations supplémentaires

- Pour des exemples complets des extraits de code ci-dessus, voir [Travailler avec Amazon DynamoDB](#), [Travailler avec Amazon EC2](#) et [Travailler avec Amazon S3](#)

Fournir des informations d'identification temporaires au SDK

Avant de demander Amazon Web Services l'utilisation du AWS SDK for Java 2.x, le SDK signe cryptographiquement les informations d'identification temporaires émises par AWS. Pour accéder aux informations d'identification temporaires, le SDK récupère les valeurs de configuration en vérifiant plusieurs emplacements.

Cette rubrique décrit plusieurs manières d'activer le SDK pour accéder aux informations d'identification temporaires.

Rubriques

- [Configuration de l'accès aux informations d'identification temporaires](#)
- [Chaîne de fournisseurs d'informations d'identification par défaut](#)
- [Utiliser un fournisseur d'informations d'identification ou une chaîne de fournisseurs spécifique](#)
- [Profils d'utilisation](#)

- [Charger des informations d'identification temporaires à partir d'un processus externe](#)
- [Fournir des informations d'identification temporaires dans le code](#)
- [Lisez les informations d'identification du rôle IAM sur Amazon EC2](#)

Configuration de l'accès aux informations d'identification temporaires

Pour une sécurité accrue, il est AWS recommandé de configurer le SDK pour Java de manière [à utiliser des informations d'identification temporaires](#) plutôt que des informations d'identification de longue durée. Les informations d'identification temporaires se composent de clés d'accès (identifiant de clé d'accès et clé d'accès secrète) et d'un jeton de session. Nous vous recommandons de [configurer le SDK](#) pour obtenir automatiquement des informations d'identification temporaires, étant donné que le processus d'actualisation des jetons est automatique. Vous pouvez toutefois [fournir directement au SDK des informations d'identification temporaires](#).

Configuration du centre d'identité IAM

Lorsque vous configurez le SDK pour utiliser l'accès par authentification unique à IAM Identity Center, comme décrit [???](#) dans ce guide, le SDK utilise automatiquement des informations d'identification temporaires.

Le SDK utilise le jeton d'accès IAM Identity Center pour accéder au rôle IAM configuré avec le `sso_role_name` paramètre de votre fichier. `config` Le SDK assume ce rôle IAM et récupère les informations d'identification temporaires à utiliser pour les demandes. Service AWS

Pour plus de détails sur la manière dont le SDK obtient des informations d'identification temporaires à partir de la configuration, consultez la section [Comprendre l'authentification IAM Identity Center](#) du Guide de référence des AWS SDK et des outils.

Récupérer depuis le portail AWS d'accès

Comme alternative à la configuration d'authentification unique d'IAM Identity Center, vous pouvez copier et utiliser les informations d'identification temporaires disponibles sur le portail d' AWS accès. Vous pouvez utiliser les informations d'identification temporaires dans un profil ou les utiliser comme valeurs pour les propriétés du système et les variables d'environnement.

Configuration d'un fichier d'informations d'identification local pour les informations d'identification temporaires

1. [Création d'un fichier d'informations d'identification partagé](#)

Chaîne de fournisseurs d'informations d'identification par défaut

La chaîne de fournisseurs d'informations d'identification par défaut est implémentée par la [DefaultCredentialsProvider](#) classe. Il vérifie séquentiellement chaque endroit où vous pouvez définir la configuration par défaut pour fournir des informations d'identification temporaires, puis sélectionne la première que vous définissez.

Pour utiliser la chaîne de fournisseurs d'informations d'identification par défaut pour fournir des informations d'identification temporaires, créez un générateur de clients de services, mais ne spécifiez pas de fournisseur d'informations d'identification. L'extrait de code suivant crée un fichier `DynamoDbClient` qui utilise la chaîne de fournisseurs d'informations d'identification par défaut pour localiser et récupérer les paramètres de configuration par défaut.

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb =
    DynamoDbClient.builder()
        .region(region)
        .build();
```

Ordre de récupération des paramètres d'identification

La chaîne de fournisseurs d'informations d'identification par défaut du SDK for Java 2.x recherche la configuration dans votre environnement à l'aide d'une séquence prédéfinie.

1. Propriétés du système Java

- Le SDK utilise la [SystemPropertyCredentialsProvider](#) classe pour charger des informations d'identification temporaires à partir des propriétés système `aws.accessKeyId`, `aws.secretAccessKey`, et `aws.sessionToken` Java.

Note

Pour plus d'informations sur la façon de définir les propriétés du système Java, consultez le didacticiel sur les [propriétés du système](#) sur le site Web officiel des didacticiels Java.

2. Variables d'environnement

- Le SDK utilise la [EnvironmentVariableCredentialsProvider](#) classe pour charger des informations d'identification temporaires à partir des variables d'environnement `AWS_SESSION_TOKEN`, `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, et.

3. Jeton d'identité Web de AWS Security Token Service

- Le SDK utilise la [WebIdentityTokenFileCredentialsProvider](#) classe pour charger des informations d'identification temporaires à partir des propriétés du système Java ou des variables d'environnement.

4. Le partage `credentials` et les `config` fichiers

- Le SDK utilise le [ProfileCredentialsProvider](#) pour charger les paramètres d'authentification unique ou les informations d'identification temporaires d'IAM Identity Center à partir du `[default]` profil dans les fichiers partagés `credentials` et `config`

Le guide de référence AWS des SDK et des outils contient [des informations détaillées](#) sur la façon dont le SDK for Java fonctionne avec le jeton d'authentification unique IAM Identity Center pour obtenir des informations d'identification temporaires que le SDK utilise pour appeler Services AWS

Note

Les `config` fichiers `credentials` et sont partagés par différents AWS SDK et outils. Pour plus d'informations, consultez [les fichiers `.aws/credentials` et `.aws/config`](#) dans le Guide de référence des SDK et des AWS outils.

5. Amazon ECS informations d'identification du conteneur

- Le SDK utilise la [ContainerCredentialsProvider](#) classe pour charger des informations d'identification temporaires à partir de la variable `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` d'environnement système.

6. Amazon EC2 informations d'identification fournies par le rôle IAM de l'instance

- Le SDK utilise la [InstanceProfileCredentialsProvider](#) classe pour charger des informations d'identification temporaires à partir du service de Amazon EC2 métadonnées.

Utiliser un fournisseur d'informations d'identification ou une chaîne de fournisseurs spécifique

Comme alternative à la chaîne de fournisseurs d'informations d'identification par défaut, vous pouvez spécifier le fournisseur d'informations d'identification que le SDK doit utiliser. Lorsque vous fournissez un fournisseur d'informations d'identification spécifique, le SDK ignore le processus de vérification

des différents emplacements, ce qui réduit légèrement le temps nécessaire à la création d'un client de service.

Par exemple, si vous définissez votre configuration par défaut à l'aide de variables d'environnement, fournissez un [EnvironmentVariableCredentialsProvider](#) objet à la `credentialsProvider` méthode dans le générateur de clients de services, comme dans l'extrait de code suivant.

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .build();
```

Pour une liste complète des fournisseurs d'informations d'identification et des chaînes de fournisseurs, voir Toutes les classes d'implémentation connues dans [AwsCredentialsProvider](#).

Note

Vous pouvez utiliser votre propre fournisseur d'informations d'identification ou des chaînes de fournisseurs en implémentant l'`AwsCredentialsProvider` interface.

Profils d'utilisation

À l'aide du `credentials` fichier partagé `config` et du fichier, vous pouvez configurer plusieurs profils. Cela permet à votre application d'utiliser plusieurs ensembles de configuration d'informations d'identification. Le `[default]` profil a été mentionné précédemment. Le SDK utilise la [ProfileCredentialsProvider](#) classe pour charger les paramètres à partir des profils définis dans le `credentials` fichier partagé.

L'extrait de code suivant montre comment créer un client de service qui utilise les paramètres définis dans le cadre du profil nommé `my_profile`

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create("my_profile"))
    .build();
```

Définir un profil différent comme profil par défaut

Pour définir un profil autre que le [default] profil par défaut pour votre application, définissez la variable d'AWS_PROFILE environnement sur le nom de votre profil personnalisé.

Pour définir cette variable sous Linux, macOS ou Unix, utilisez `export` :

```
export AWS_PROFILE="other_profile"
```

Pour définir ces variables sous Windows, utilisez `set` :

```
set AWS_PROFILE="other_profile"
```

Vous pouvez également définir la propriété système `aws.profile` Java sur le nom du profil.

Recharger les informations d'identification du profil

Vous pouvez configurer n'importe quel fournisseur d'informations d'identification dont le générateur dispose d'une `profileFile()` méthode permettant de recharger les informations d'identification du profil. Ces classes de profil d'identification sont les suivantes : `ProfileCredentialsProviderDefaultCredentialsProvider`, `InstanceProfileCredentialsProvider` et `ProfileTokenProvider`.

Note

Le rechargement des informations d'identification du profil ne fonctionne qu'avec les paramètres suivants du fichier de profil : `aws_access_key_id`, `aws_secret_access_key`, et `aws_session_token`

Les paramètres tels que `region`, `sso_session`, `sso_account_id`, et `source_profile` sont ignorés.

Pour configurer un fournisseur d'informations d'identification pris en charge afin de recharger les paramètres de profil, fournissez une instance [ProfileFileSupplier](#) de la méthode `profileFile()` Builder. L'exemple de code suivant illustre un système `ProfileCredentialsProvider` qui recharge les paramètres d'identification à partir du [default] profil.

```

ProfileCredentialsProvider provider = ProfileCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.defaultSupplier())
    .build();

// Set up a service client with the provider instance.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(provider)
    .build();

/*
    Before dynamoDbClient makes a request, it reloads the credentials settings
    by calling provider.resolveCredentials().
*/

```

Lorsqu'il `ProfileCredentialsProvider.resolveCredentials()` est appelé, le SDK for Java recharge les paramètres. `ProfileFileSupplier.defaultSupplier()` est l'une des [nombreuses implémentations pratiques](#) `ProfileFileSupplier` fournies par le SDK. Si votre cas d'utilisation l'exige, vous pouvez fournir votre propre implémentation.

L'exemple suivant montre l'utilisation de la méthode `ProfileFileSupplier.reloadWhenModified()` pratique. `reloadWhenModified()` prend un `Path` paramètre, ce qui vous permet de désigner le fichier source pour la configuration plutôt que l'emplacement standard `~/.aws/credentials` (ou `ouconfig`).

Les paramètres seront rechargés lors `resolveCredentials()` de l'appel uniquement si le SDK détermine que le contenu du fichier a été modifié.

```

Path credentialsFilePath = ...

ProfileCredentialsProvider provider = ProfileCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
        ProfileFile.Type.CREDENTIALS))
    .profileName("my-profile")
    .build();

/*
    A service client configured with the provider instance calls
    provider.resolveCredential()
    before each request.
*/

```

```
*/
```

Le `ProfileFileSupplier.aggregate()` procédé fusionne le contenu de plusieurs fichiers de configuration. Vous décidez si un fichier est rechargé par appel à `resolveCredentials()` ou si les paramètres d'un fichier sont fixés au moment de sa première lecture.

L'exemple suivant montre un `DefaultCredentialsProvider` qui fusionne les paramètres de deux fichiers contenant des paramètres de profil. Le SDK recharge les paramètres dans le fichier pointé par la `credentialsFilePath` variable chaque fois qu'`resolveCredentials()` il est appelé et que les paramètres sont modifiés. Les paramètres de l'`profileFile` objet restent les mêmes.

```
Path credentialsFilePath = ...;
ProfileFile profileFile = ...;

DefaultCredentialsProvider provider = DefaultCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.aggregate(
        ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
        ProfileFile.Type.CREDENTIALS),
        ProfileFileSupplier.fixedProfileFile(profileFile)))
    .profileName("my-profile")
    .build();
/*
   A service client configured with the provider instance calls
   provider.resolveCredential()
   before each request.
*/
```

Charger des informations d'identification temporaires à partir d'un processus externe

Warning

Ce qui suit décrit une méthode d'obtention d'informations d'identification temporaires auprès d'un processus externe. Cela peut être dangereux, alors soyez prudent. Dans la mesure du possible, il convient de privilégier d'autres fournisseurs d'informations d'identification. Si vous utilisez cette option, vous devez vous assurer que le `config` fichier est aussi verrouillé que possible conformément aux meilleures pratiques de sécurité de votre système d'exploitation.

Assurez-vous que votre outil d'identification personnalisé n'écrit aucune information secrète dans `StdErr`. Les SDK AWS CLI peuvent également capturer et enregistrer ces informations, les exposant potentiellement à des utilisateurs non autorisés.

Avec le SDK pour Java 2.x, vous pouvez obtenir des informations d'identification temporaires auprès d'un processus externe pour des cas d'utilisation personnalisés. Il existe deux manières de configurer cette fonctionnalité.

Utiliser le **credential_process** réglage

Si vous disposez d'une méthode qui fournit des informations d'identification temporaires, vous pouvez l'intégrer en ajoutant le `credential_process` paramètre dans le cadre d'une définition de profil dans le config fichier. La valeur que vous spécifiez doit utiliser le chemin complet du fichier de commandes. Si le chemin du fichier contient des espaces, vous devez l'entourer de guillemets.

Le SDK appelle la commande exactement comme indiqué, puis lit les données JSON depuis `stdout`.

Les exemples suivants montrent l'utilisation de ce paramètre pour les chemins de fichiers sans espaces et les chemins de fichiers contenant des espaces.

Linux/macOS

Aucun espace dans le chemin du fichier

```
[profile process-credential-profile]
credential_process = /path/to/credential/file/credential_file.sh --custom-command
custom_parameter
```

Espaces dans le chemin du fichier

```
[profile process-credential-profile]
credential_process = "/path/with/space to/credential/file/credential_file.sh" --
custom-command custom_parameter
```

Windows

Aucun espace dans le chemin du fichier

```
[profile process-credential-profile]
```

```
credential_process = C:\Path\To\credentials.cmd --custom_command custom_parameter
```

Espaces dans le chemin du fichier

```
[profile process-credential-profile]
credential_process = "C:\Path\With Space To\credentials.cmd" --custom_command
custom_parameter
```

L'extrait de code suivant montre comment créer un client de service qui utilise les informations d'identification temporaires définies dans le cadre du profil nommé `process-credential-profile`

```
Region region = Region.US_WEST_2;
S3Client s3Client = S3Client.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create("process-credential-
profile"))
    .build();
```

Pour obtenir des informations détaillées sur l'utilisation d'un processus externe comme source d'informations d'identification temporaires, reportez-vous à la [section relative aux informations d'identification du processus](#) dans le Guide de référence AWS des SDK et des outils.

Utilisation de la **ProcessCredentialsProvider**

Au lieu d'utiliser les paramètres du config fichier, vous pouvez utiliser le SDK [ProcessCredentialsProvider](#) pour charger des informations d'identification temporaires à l'aide de Java.

Les exemples suivants montrent différentes versions de la manière de spécifier un processus externe à l'aide des informations d'identification temporaires `ProcessCredentialsProvider` et de la configuration d'un client de service utilisant les informations d'identification temporaires.

Linux/macOS

Aucun espace dans le chemin du fichier

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
```

```

        .command("/path/to/credentials.sh optional_param1 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(credentials)
    .build();

```

Espaces dans le chemin du fichier

```

ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("/path\\ with\\ spaces\\ to/credentials.sh optional_param1
optional_param2")
        .build();

S3Client s3 = S3Client.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(credentials)
    .build();

```

Windows

Aucun espace dans le chemin du fichier

```

ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("C:\\Path\\To\\credentials.exe optional_param1 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(credentials)
    .build();

```

Espaces dans le chemin du fichier

```

ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()

```

```
        .command("\"C:\\Path\\With Spaces To\\credentials.exe\" optional_param1  
optional_param2")  
        .build();  
  
S3Client s3 = S3Client.builder()  
    .region(Region.US_WEST_2)  
    .credentialsProvider(credentials)  
    .build();
```

Fournir des informations d'identification temporaires dans le code

Si la chaîne d'informations d'identification par défaut ou un fournisseur ou une chaîne de fournisseurs spécifique ou personnalisée ne fonctionnent pas pour votre application, vous pouvez fournir des informations d'identification temporaires directement dans le code. Il peut s'agir [d'informations d'identification de rôle IAM décrites ci-dessus](#) ou d'informations d'identification temporaires extraites de AWS Security Token Service (AWS STS). Si vous avez récupéré des informations d'identification temporaires à l'aide de AWS STS, fournissez-les à un Service AWS client comme indiqué dans l'exemple de code suivant.

1. Assumez un rôle en appelant `StsClient.assumeRole()`.
2. Créez un [StaticCredentialsProvider](#) objet et fournissez-le avec l'`AwsSessionCredentials` objet.
3. Configurez le générateur de clients de service avec `StaticCredentialsProvider` et créez le client.

L'exemple suivant crée un client de service Amazon S3 à l'aide d'informations d'identification temporaires renvoyées par AWS STS pour un rôle assumé par IAM.

```
// The AWS IAM Identity Center identity (user) who executes this method does not  
// have permission to list buckets.  
// The identity is configured in the [default] profile.  
public static void assumeRole(String roleArn, String roleSessionName) {  
    // The IAM role represented by the 'roleArn' parameter can be assumed by  
    // identities in two different accounts  
    // and the role permits the user to only list buckets.  
  
    // The SDK's default credentials provider chain will find the single sign-on  
    // settings in the [default] profile.  
    // The identity configured with the [default] profile needs permission to call  
    // AssumeRole on the STS service.
```



```
try {
    Credentials tempRoleCredentials;
    try (StsClient stsClient = StsClient.create()) {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        tempRoleCredentials = roleResponse.credentials();
    }
    // Use the following temporary credential items for the S3 client.
    String key = tempRoleCredentials.accessKeyId();
    String secKey = tempRoleCredentials.secretAccessKey();
    String secToken = tempRoleCredentials.sessionToken();

    // List all buckets in the account associated with the assumed role
    // by using the temporary credentials retrieved by invoking
    stsClient.assumeRole().
        StaticCredentialsProvider staticCredentialsProvider =
        StaticCredentialsProvider.create(
            AwsSessionCredentials.create(key, secKey, secToken));
    try (S3Client s3 = S3Client.builder()
        .credentialsProvider(staticCredentialsProvider)
        .build()) {
        List<Bucket> buckets = s3.listBuckets().buckets();
        for (Bucket bucket : buckets) {
            System.out.println("bucket name: " + bucket.name());
        }
    }
} catch (StsException | S3Exception e) {
    logger.error(e.getMessage());
    System.exit(1);
}
}
```

Ensemble d'autorisations

L'ensemble d'autorisations suivant défini dans AWS IAM Identity Center permet à l'identité (utilisateur) d'effectuer les deux opérations suivantes

1. GetObjectFonctionnement du service Amazon Simple Storage.
2. Le AssumeRole fonctionnement du AWS Security Token Service.

Si vous n'assumez pas le rôle, la `s3.listBuckets()` méthode illustrée dans l'exemple échouerait.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "sts:AssumeRole"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Rôle endossé

Politique d'autorisation des rôles assumés

La politique d'autorisation suivante est attachée au rôle assumé dans l'exemple précédent. Cette politique d'autorisation permet de répertorier tous les buckets dans le même compte que le rôle.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Politique de confiance fondée sur les rôles assumés

La politique de confiance suivante est attachée au rôle assumé dans l'exemple précédent. La politique permet aux identités (utilisateurs) d'assumer le rôle dans deux comptes.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:root",
          "arn:aws:iam::555555555555:root"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

Lisez les informations d'identification du rôle IAM 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 AWS CLI d' AWS API. 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.

Cette rubrique fournit des informations sur la façon de configurer votre application Java pour qu'elle s'exécute sur une instance EC2 et de permettre au SDK pour Java d' IAM acquérir des informations d'identification de rôle.

Acquérir les informations d'identification des rôles IAM auprès de l'environnement

Si votre application crée un client de AWS service à l'aide de la `create` méthode (ou `builder().build()` des méthodes), le SDK for Java utilise la chaîne de fournisseurs d'informations d'identification par défaut. La chaîne de fournisseurs d'informations d'identification par défaut recherche dans l'environnement d'exécution des éléments de configuration que le SDK peut échanger contre des informations d'identification temporaires. La [the section called “Chaîne de fournisseurs d'informations d'identification par défaut”](#) section décrit le processus de recherche complet.

La dernière étape de la chaîne de fournisseurs par défaut n'est disponible que lorsque votre application s'exécute sur une Amazon EC2 instance. Au cours de cette étape, le SDK utilise un `InstanceProfileCredentialsProvider` pour lire le rôle IAM défini dans le profil d'instance EC2. Le SDK acquiert ensuite des informations d'identification temporaires pour ce rôle IAM.

Bien que ces informations d'identification soient temporaires et finiront par expirer, elles sont `InstanceProfileCredentialsProvider` régulièrement actualisées pour vous afin qu'elles continuent à autoriser l'accès à AWS.

Acquérir les informations d'identification des rôles IAM par programmation

Comme alternative à la chaîne de fournisseurs d'informations d'identification par défaut qui utilise éventuellement un `InstanceProfileCredentialsProvider` sur EC2, vous pouvez configurer un client de service de manière explicite avec un `InstanceProfileCredentialsProvider`. Cette approche est illustrée dans l'extrait suivant.

```
S3Client s3 = S3Client.builder()
    .credentialsProvider(InstanceProfileCredentialsProvider.create())
    .build();
```

Acquérir des informations d'identification de rôle IAM en toute

Par défaut, les instances EC2 exécutent le service [IMDS](#) (Instance Metadata Service) qui permet aux SDK d'accéder `InstanceProfileCredentialsProvider` à des informations telles que le rôle IAM configuré. Les instances EC2 exécutent deux versions d'IMDS par défaut :

- Service des métadonnées d'instance Version 1 (IMDSv1) – méthode de demande/réponse
- Service des métadonnées d'instance Version 2 (IMDSv2) – méthode orientée session

[IMDSv2 est une approche plus sécurisée](#) que IMDSv1.

Par défaut, le SDK Java essaie d'abord IMDSv2 pour obtenir le rôle IAM, mais en cas d'échec, il essaie IMDSv1. Cependant, comme IMDSv1 est moins sécurisé, il est AWS recommandé d'utiliser uniquement IMDSv2 et d'empêcher le SDK d'essayer IMDSv1.

Pour utiliser l'approche la plus sécurisée, empêchez le SDK d'utiliser IMDSv1 en fournissant l'un des paramètres suivants avec une valeur de `true`

- Variable d'environnement : `AWS_EC2_METADATA_V1_DISABLED`
- Propriété du système JVM : `aws.disableEc2MetadataV1`
- Paramètre du fichier de configuration partagé : `ec2_metadata_v1_disabled`

Lorsque l'un de ces paramètres est défini sur `true`, le SDK ne charge pas les informations d'identification du rôle IMDS en utilisant IMDSv1 en cas d'échec de l'appel IMDSv2 initial.

Utiliser Régions AWS

Régions AWS permettre aux clients du service d'accéder à ceux Services AWS qui se trouvent physiquement dans une zone géographique spécifique.

Configurez explicitement un Région AWS

Pour définir explicitement une région, nous vous recommandons d'utiliser les constantes définies dans la classe [Region](#). Il s'agit d'une énumération de toutes les régions disponibles publiquement.

Pour créer un client avec une région énumérée à partir de la classe, utilisez la `region` méthode du générateur de clients.

```
Ec2Client ec2 = Ec2Client.builder()
    .region(Region.US_WEST_2)
    .build();
```

Si la région que vous souhaitez utiliser n'est pas l'une des énumérations de la `Region` classe, vous pouvez créer une nouvelle région à l'aide de la méthode statique `of`. Cette méthode vous permet d'accéder à de nouvelles régions sans mettre à niveau le SDK.

```
Region newRegion = Region.of("us-east-42");
```

```
Ec2Client ec2 = Ec2Client.builder()
    .region(newRegion)
    .build();
```

Note

Une fois que vous avez créé un client avec le générateur, il est immuable et Région AWS ne peut pas être modifié. Si vous devez travailler avec plusieurs clients Régions AWS pour le même service, vous devez créer plusieurs clients, un par région.

Laissez le SDK déterminer automatiquement la région à partir de l'environnement

Lorsque votre code s'exécute sur Amazon EC2 ou AWS Lambda, vous souhaitez peut-être configurer les clients pour qu'ils utilisent le même système Région AWS que celui sur lequel votre code s'exécute. Cela dissocie votre code de l'environnement dans lequel il s'exécute et facilite le déploiement de votre application sur plusieurs Régions AWS sites afin de réduire la latence ou la redondance.

Pour utiliser la chaîne de fournisseurs d'informations d'identification/région par défaut afin de déterminer la région à partir de l'environnement, utilisez la méthode du générateur de `create` clients.

```
Ec2Client ec2 = Ec2Client.create();
```

Si vous ne définissez pas explicitement un Région AWS en utilisant la `region` méthode, le SDK consulte la chaîne de fournisseurs de régions par défaut pour déterminer la région à utiliser.

Comprendre la chaîne de fournisseurs de régions par défaut

Le SDK suit les étapes suivantes pour rechercher un Région AWS :

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

Note

Le Lambda conteneur définit cette variable d'environnement.

3. Le SDK vérifie le fichier de configuration AWS partagé et le fichier d'informations d'identification partagé (généralement situés à l'emplacement `~/.aws/config` et `~/.aws/credentials`). Si la propriété `region` est présente, le SDK l'utilise.
 - Si le SDK trouve la propriété `region` dans les deux fichiers pour le même profil (y compris le profil `default`), il utilise la valeur du fichier d'informations d'identification partagé.
 - 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 spécifier le profil chargé par le SDK.
4. Le SDK tente d'utiliser le service de métadonnées d'Amazon EC2 instance (IMDS) pour déterminer la région de l'instance en cours d'exécution Amazon EC2 .
 - Pour plus de sécurité, vous devez empêcher le SDK d'essayer d'utiliser la version 1 d'IMDS. Pour désactiver la version 1, vous utilisez le même paramètre que celui décrit dans la [section appelée "En toute sécurité"](#) section.
5. Si le SDK n'a toujours pas trouvé de région à ce stade, la création du client échoue avec une exception.

Lors du développement d'AWS applications, une approche courante consiste à utiliser le fichier de configuration partagé (décrit dans [Ordre de récupération des informations d'identification](#)) 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 lorsque l'application s'exécute sur AWS l'infrastructure. La création du client s'en trouve ainsi grandement simplifiée et votre application demeure portable.

Vérifier la disponibilité du service dans une région

Pour voir si un produit particulier Service AWS est disponible dans une région, utilisez la `region` méthode `serviceMetadata` et sur le client du service.

```
DynamoDbClient.serviceMetadata().regions().forEach(System.out::println);
```

Consultez la documentation de la classe [Region](#) pour savoir ce Régions AWS que vous pouvez spécifier et utilisez le préfixe de point de terminaison du service pour effectuer une requête.

Choisissez un point de terminaison spécifique

Dans certaines situations, par exemple pour tester les fonctionnalités d'aperçu d'un service avant que les fonctionnalités ne soient mises à disposition générale, vous devrez peut-être spécifier un point de terminaison spécifique dans une région. Dans ces situations, les clients du service peuvent être configurés en appelant la `endpointOverride` méthode.

Par exemple, pour configurer un Amazon EC2 client afin qu'il utilise la région Europe (Irlande) avec un point de terminaison spécifique, utilisez le code suivant.

```
Ec2Client ec2 = Ec2Client.builder()
    .region(Region.EU_WEST_1)
    .endpointOverride(URI.create("https://ec2.eu-west-1.amazonaws.com"))
    .build();
```

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

Réduisez le temps de démarrage du SDK pour AWS Lambda

L'un des objectifs du AWS SDK for Java 2.x est de réduire la latence de démarrage des AWS Lambda fonctions. Le SDK contient des modifications qui réduisent le temps de démarrage, qui sont abordées à la fin de cette rubrique.

Tout d'abord, cette rubrique se concentre sur les modifications que vous pouvez apporter pour réduire les temps de démarrage à froid. Il s'agit notamment de modifier la structure de votre code et la configuration des clients de service.

Utiliser un client AWS HTTP CRT

Pour travailler avec AWS Lambda, nous recommandons l'utilisation [AwsCrtHttpClient](#) pour les scénarios synchrones et celle [AwsCrtAsyncHttpClient](#) pour les scénarios asynchrones.

La [the section called “Configuration de clients AWS HTTP basés sur CRT”](#) rubrique de ce guide décrit les avantages de l'utilisation des clients HTTP, comment ajouter la dépendance et comment configurer leur utilisation par les clients de service.

Supprimer les dépendances du client HTTP inutilisées

Outre l'utilisation explicite d'un client AWS CRT, vous pouvez supprimer d'autres clients HTTP introduits par défaut par le SDK. Le temps de démarrage de Lambda est réduit lorsque moins de bibliothèques doivent être chargées. Vous devez donc supprimer tous les artefacts inutilisés que la machine virtuelle Java doit charger.

L'extrait de `pom.xml` fichier Maven suivant montre l'exclusion du client HTTP basé sur Apache et du client HTTP basé sur Netty. (Ces clients ne sont pas nécessaires lorsque vous utilisez un client AWS CRT.) Cet exemple exclut les artefacts du client HTTP de la dépendance du client S3 et ajoute `aws-crt-client` pour autoriser l'accès aux clients HTTP AWS basés sur CRT.

```
<project>
  <properties>
    <aws.java.sdk.version>2.25.51</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
      <exclusions>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>netty-nio-client</artifactId>
        </exclusion>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
```

```
        <artifactId>apache-client</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
</dependencies>
</project>
```

Note

Ajoutez l'<exclusions>élément à toutes les dépendances des clients de service dans votre pom.xml fichier.

Configuration des clients de service pour des recherches de raccourcis

Spécifiez une région

Lorsque vous créez un client de service, appelez la `region` méthode dans le générateur de clients de services. Cela permet de raccourcir le [processus de recherche de région](#) par défaut du SDK, qui vérifie les Région AWS informations à plusieurs endroits.

Pour que le code Lambda reste indépendant de la région, utilisez le code suivant dans la `region` méthode. Ce code accède à la variable d'AWS_REGION environnement définie par le conteneur Lambda.

```
Region.of(System.getenv(SdkSystemSetting.AWS_REGION.environmentVariable()))
```

Utilisation de la **EnvironmentVariableCredentialProvider**

Tout comme le comportement de recherche par défaut pour les informations de région, le SDK recherche les informations d'identification à plusieurs endroits. En spécifiant le [EnvironmentVariableCredentialProvider](#) moment où vous créez un client de service, vous gagnez du temps dans le processus de recherche du SDK.

Note

L'utilisation de ce fournisseur d'informations d'identification permet d'utiliser le code dans Lambda les fonctions, mais risque de ne pas fonctionner sur Amazon EC2 d'autres systèmes.

L'extrait de code suivant montre un client de service S3 correctement configuré pour être utilisé dans un environnement Lambda.

```
S3Client s3Client = S3Client.builder()

    .region(Region.of(System.getenv(SdkSystemSetting.AWS_REGION.environmentVariable())))
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .httpClient(AwsCrtHttpClient.builder().build())
    .build();
```

Initialisez le client SDK en dehors du gestionnaire de fonctions Lambda

Nous recommandons d'initialiser un client SDK en dehors de la méthode du gestionnaire Lambda. Ainsi, si le contexte d'exécution est réutilisé, l'initialisation du client de service peut être ignorée. En réutilisant l'instance cliente et ses connexions, les appels ultérieurs de la méthode du gestionnaire se produisent plus rapidement.

Dans l'exemple suivant, l'`S3Client` instance est initialisée dans le constructeur à l'aide d'une méthode d'usine statique. Si le conteneur géré par l'environnement Lambda est réutilisé, l'instance initialisée `S3Client` est réutilisée.

```
public class App implements RequestHandler<Object, Object> {
    private final S3Client s3Client;

    public App() {
        s3Client = DependencyFactory.s3Client();
    }

    @Override
    public Object handle Request(final Object input, final Context context) {
        ListBucketResponse response = s3Client.listBuckets();
        // Process the response.
    }
}
```

Minimiser l'injection de dépendance

Les frameworks d'injection de dépendances (DI) peuvent prendre plus de temps pour terminer le processus de configuration. Ils peuvent également nécessiter des dépendances supplémentaires, dont le chargement prend du temps.

Si un framework DI est nécessaire, nous vous recommandons d'utiliser des frameworks DI légers tels que [Dagger](#).

Utilisez un archétype de ciblage Maven AWS Lambda

L'équipe du SDK AWS Java a développé un modèle [Maven Archetype](#) pour démarrer un projet Lambda avec un temps de démarrage minimal. Vous pouvez créer un projet Maven à partir de l'archétype et savoir que les dépendances sont configurées de manière appropriée pour l'environnement Lambda.

Pour en savoir plus sur l'archétype et travailler sur un exemple de déploiement, consultez ce [billet de blog](#).

Pensez à Lambda SnapStart pour Java

Si vos exigences d'exécution sont compatibles, AWS [Lambda SnapStart pour Java](#) est disponible. Lambda SnapStart est une solution basée sur l'infrastructure qui améliore les performances de démarrage des fonctions Java. Lorsque vous publiez une nouvelle version d'une fonction, Lambda l'initialise et prend un instantané chiffré immuable de l'état de la mémoire et du disque. SnapStart met ensuite en cache l'instantané pour le réutiliser.

Modifications de la version 2.x qui affectent le temps de démarrage

Outre les modifications que vous apportez à votre code, la version 2.x du SDK pour Java inclut trois modifications principales qui réduisent le temps de démarrage :

- Utilisation de [jackson-jr](#), une bibliothèque de sérialisation qui améliore le temps d'initialisation
- Utilisation des bibliothèques [java.time](#) pour les objets de date et d'heure, qui font partie du JDK
- Utilisation du [SLF4j](#) pour une façade en bois

Ressources supplémentaires

Le Guide du AWS Lambda développeur contient une [section sur les meilleures pratiques](#) pour le développement de fonctions Lambda qui n'est pas spécifique à Java.

Pour un exemple de création d'une application native pour le cloud en Java qui utilise AWS Lambda, consultez le [contenu de cet atelier](#). L'atelier traite de l'optimisation des performances et d'autres meilleures pratiques.

Vous pouvez envisager d'utiliser des images statiques compilées à l'avance pour réduire le temps de latence au démarrage. Par exemple, vous pouvez utiliser le SDK pour Java 2.x et Maven pour [créer une image native de GraalVM](#).

Clients HTTP

Vous pouvez modifier le client HTTP à utiliser pour votre client de service et modifier la configuration par défaut pour les clients HTTP avec le AWS SDK for Java 2.x. Cette section décrit les clients HTTP et les paramètres du SDK.

Clients HTTP disponibles dans le SDK for Java

Clients synchrones

Les clients HTTP synchrones du SDK for Java [SdkHttpClient](#) implémentent l'interface. Un client de service synchrone, tel que le `S3Client` ou le `DynamoDbClient`, nécessite l'utilisation d'un client HTTP synchrone. AWS SDK for Java II propose trois clients HTTP synchrones.

ApacheHttpClient (par défaut)

[ApacheHttpClient](#) est le client HTTP par défaut pour les clients de service synchrones. Pour plus d'informations sur la configuration du `ApacheHttpClient`, consultez [Configuration du client HTTP basé sur Apache](#).

AwsCrtHttpClient

[AwsCrtHttpClient](#) fournit un débit élevé et des E/S non bloquantes. Il est basé sur le client HTTP AWS Common Runtime (CRT). Pour plus d'informations sur la configuration du `AwsCrtHttpClient` et son utilisation avec les clients de service, consultez [the section called "Configuration de clients AWS HTTP basés sur CRT"](#).

URLConnectionHttpClient

Pour minimiser le nombre de fichiers JAR et de bibliothèques tierces utilisés par votre application, vous pouvez utiliser le [URLConnectionHttpClient](#). Pour plus d'informations sur la configuration du `URLConnectionHttpClient`, consultez [Configuration du client HTTP basé sur une connexion URL](#).

Clients asynchrones

Les clients HTTP asynchrones du SDK for Java implémentent l'interface. [SdkAsyncHttpClient](#)

Un client de service asynchrone, tel que le `S3AsyncClient` ou le `DynamoDbAsyncClient`, nécessite l'utilisation d'un client HTTP asynchrone. AWS SDK for Java II propose deux clients HTTP asynchrones.

`NettyNioAsyncHttpClient` (par défaut)

[NettyNioAsyncHttpClient](#) est le client HTTP par défaut utilisé par les clients asynchrones. Pour plus d'informations sur la configuration du `NettyNioAsyncHttpClient`, consultez [the section called "Configuration du client HTTP basé sur Netty"](#).

`AwsCrtAsyncHttpClient`

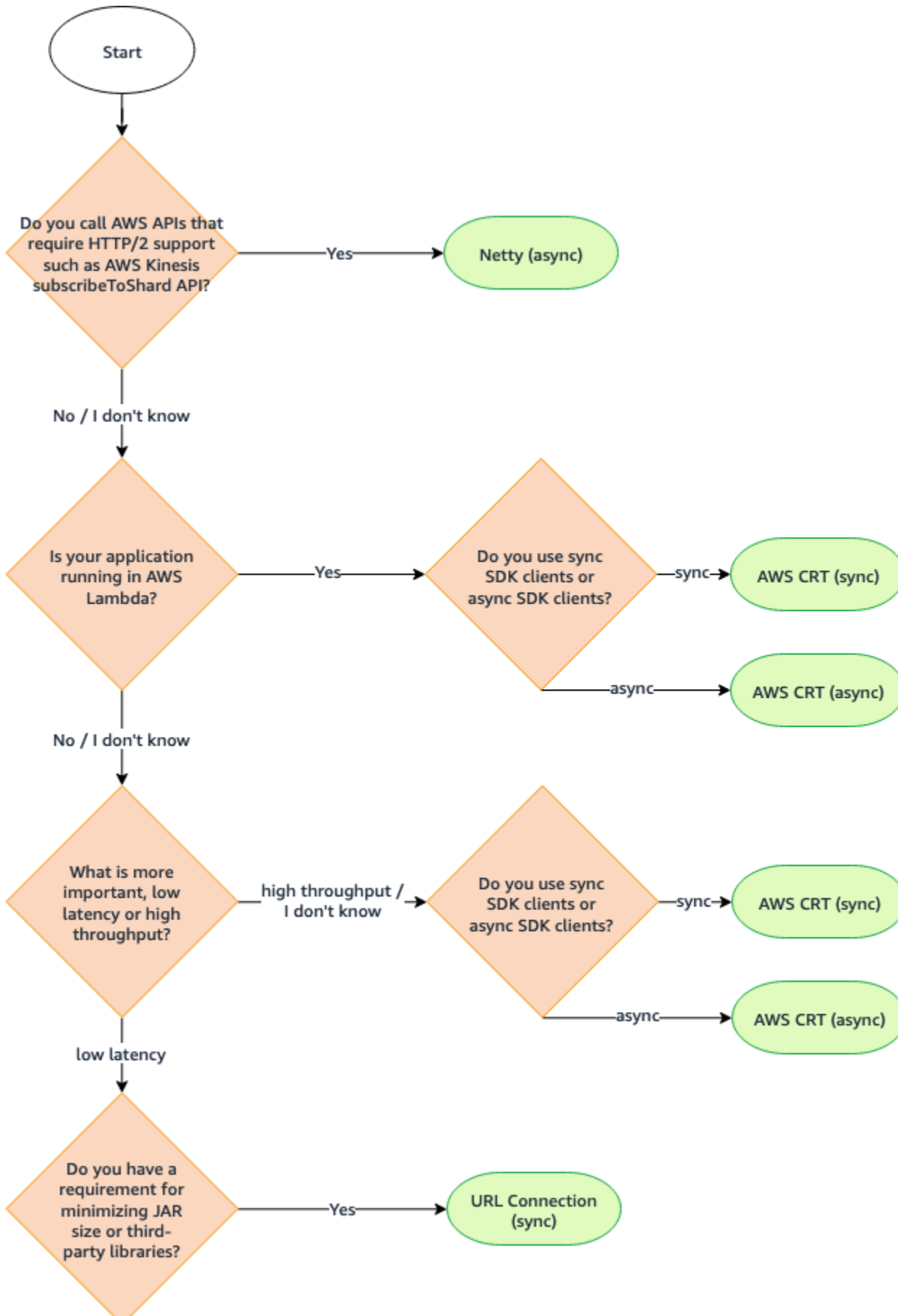
[AwsCrtAsyncHttpClient](#) est basé sur le client HTTP AWS Common Runtime (CRT). Pour plus d'informations sur la configuration du `AwsCrtAsyncHttpClient`, consultez [the section called "Configuration de clients AWS HTTP basés sur CRT"](#).

Recommandations relatives aux clients HTTP

Plusieurs facteurs entrent en jeu lorsque vous choisissez une implémentation de client HTTP. Utilisez les informations suivantes pour vous aider à prendre une décision.

Organigramme des recommandations

L'organigramme suivant fournit des conseils généraux pour vous aider à déterminer le client HTTP à utiliser.



Comparaison des clients HTTP

Le tableau suivant fournit des informations détaillées pour chaque client HTTP.

Client HTTP	Synchronisation ou asynchrone	Utilisation	Limitation/ inconvénient
Client HTTP basé sur Apache (client HTTP de synchronisation par défaut)	Sync	Utilisez-le si vous préférez une faible latence à un débit élevé	Temps de démarrage plus lent par rapport aux autres clients HTTP
Client HTTP basé sur une connexion URL	Sync	Utilisez-le si vous avez des exigences strictes en matière de limitation des dépendances avec des tiers	Ne prend pas en charge la méthode HTTP PATCH, requise par certaines API telles que les opérations de mise à jour d'Amazon APIGateway
AWS Client HTTP de synchronisation basé sur CRT 1	Sync	<ul style="list-style-type: none"> • Utilisez-le si votre application s'exécute dans AWS Lambda • Utilisez-le si vous préférez un débit élevé à une faible latence • Utilisez-le si vous préférez les clients du SDK de synchronisation 	N/A
Client HTTP basé sur Netty	Asynchrone	<ul style="list-style-type: none"> • Utilisez-le si votre application invoque des API nécessitant le support HTTP/2, telles que l'API Kinesis SubscribeToShard 	Temps de démarrage plus lent par rapport aux autres clients HTTP

Client HTTP	Synchronisation ou asynchrone	Utilisation	Limitation/ inconvénient
(client HTTP asynchrone par défaut)			
AWS ^{Client HTTP} asynchrone basé sur CRT ¹	Asynchrone	<ul style="list-style-type: none"> • Utilisez-le si votre application s'exécute dans AWS Lambda • Utilisez-le si vous préférez un débit élevé à une faible latence • Utilisez-le si vous préférez les clients SDK asynchrones 	<ul style="list-style-type: none"> • Ne prend pas en charge les clients de service qui ont besoin du support HTTP/2, tels que KinesisAsyncClient, TranscribeStreamingClient

¹ En raison de leurs avantages supplémentaires, nous vous recommandons d'utiliser les clients HTTP AWS basés sur CRT si possible.

Paramètres de configuration intelligents par défaut

La AWS SDK for Java 2.x (version 2.17.102 ou ultérieure) propose une fonctionnalité de configuration intelligente par défaut. Cette fonctionnalité optimise deux propriétés du client HTTP ainsi que d'autres propriétés qui n'affectent pas le client HTTP.

Les paramètres par défaut de configuration intelligente définissent des valeurs raisonnables pour les `tlsNegotiationTimeoutInMillis` propriétés `connectTimeoutInMillis` et en fonction de la valeur du mode par défaut que vous fournissez. Vous choisissez la valeur du mode par défaut en fonction des caractéristiques de votre application.

Pour plus d'informations sur les paramètres de configuration intelligents par défaut et sur la manière de choisir la valeur de mode par défaut la mieux adaptée à vos applications, consultez le Guide de référence [AWS des SDK et des outils](#).

Voici quatre méthodes pour définir le mode par défaut de votre application.

Service client

Utilisez le générateur de client de service pour configurer le mode par défaut directement sur le client de service. L'exemple suivant définit le mode par défaut sur auto pour le `DynamoDbClient`.

```
DynamoDbClient ddbClient = DynamoDbClient.builder()
    .defaultsMode(DefaultsMode.AUTO)
    .build();
```

System property

Vous pouvez utiliser la propriété `aws.defaultsMode` système pour définir le mode par défaut. Si vous définissez la propriété système en Java, vous devez définir la propriété avant d'initialiser un client de service.

L'exemple suivant montre comment définir le mode par défaut à l'aide d'une propriété système définie en Java.

```
System.setProperty("aws.defaultsMode", "auto");
```

L'exemple suivant montre comment définir le mode par défaut à auto l'aide d'une `-D` option de la `java` commande.

```
java -Daws.defaultsMode=auto
```

Environment variable

Définissez une valeur pour la variable `AWS_DEFAULTS_MODE` d'environnement afin de sélectionner le mode par défaut pour votre application.

Les informations suivantes indiquent la commande à exécuter pour définir la valeur du mode par défaut à l'aide d'une variable d'environnement.

Système d'exploitation	Commande pour définir les variables d'environnement
Linux, macOS ou Unix	<code>export AWS_DEFAULTS_MODE=auto</code>

Système d'exploitation	Commande pour définir les variables d'environnement
Windows	<code>set AWS_DEFAULTS_MODE=auto</code>

AWS config file

Vous pouvez ajouter une propriété `defaults_mode` de configuration au AWS config fichier partagé, comme le montre l'exemple suivant.

```
[default]
defaults_mode = auto
```

Si vous définissez le mode par défaut globalement avec la propriété système, la variable d'environnement ou le fichier de AWS configuration, vous pouvez remplacer les paramètres lorsque vous créez un client HTTP.

Lorsque vous créez un client HTTP avec `httpClientBuilder()` cette méthode, les paramètres s'appliquent uniquement à l'instance que vous créez. Vous en trouverez un exemple [ici](#). Dans cet exemple, le client HTTP basé sur Netty remplace toutes les valeurs de mode par défaut définies globalement pour `et.connectTimeoutInMillis` `tlsNegotiationTimeoutInMillis`

Configuration du client HTTP basé sur Apache

Les clients de service synchrones AWS SDK for Java 2.x utilisent un client HTTP basé sur Apache, [ApacheHttpClient](#) par défaut. Le SDK `ApacheHttpClient` est basé sur Apache [HttpClient](#).

Le SDK propose également le [URLConnectionHttpClient](#), qui se charge plus rapidement, mais comporte moins de fonctionnalités. Pour plus d'informations sur la configuration du `URLConnectionHttpClient`, consultez [the section called "Configuration du client HTTP basé sur une connexion URL"](#).

Pour voir l'ensemble complet des options de configuration disponibles pour le `ApacheHttpClient`, consultez [ApacheHttpClient.Builder](#) et [ProxyConfiguration.Builder](#).

Accédez au `ApacheHttpClient`

Dans la plupart des cas, vous utilisez le `ApacheHttpClient` sans aucune configuration explicite. Vous déclarez vos clients de service et le SDK les configurera `ApacheHttpClient` avec des valeurs standard pour vous.

Si vous souhaitez le configurer explicitement `ApacheHttpClient` ou l'utiliser avec plusieurs clients de service, vous devez le rendre disponible pour la configuration.

Aucune configuration nécessaire

Lorsque vous déclarez une dépendance à l'égard d'un client de service dans Maven, le SDK ajoute une dépendance d'exécution à `apache-clientartefact`. Cela rend la `ApacheHttpClient` classe disponible pour votre code au moment de l'exécution, mais pas au moment de la compilation. Si vous ne configurez pas le client HTTP basé sur Apache, vous n'avez pas besoin de spécifier de dépendance pour celui-ci.

Dans l'extrait XML suivant d'un `pom.xml` fichier Maven, la dépendance déclarée avec `introduit <artifactId>s3</artifactId>` automatiquement le client HTTP basé sur Apache. Il n'est pas nécessaire de déclarer une dépendance spécifiquement pour celle-ci.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.17.290</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <!-- The s3 dependency automatically adds a runtime dependency on the
  ApacheHttpClient-->
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId>
  </dependency>
</dependencies>
```

Avec ces dépendances, vous ne pouvez pas apporter de modifications explicites à la configuration HTTP, car la `ApacheHttpClient` bibliothèque se trouve uniquement sur le chemin de classe d'exécution.

Configuration requise

Pour configurer le `ApacheHttpClient`, vous devez ajouter une dépendance à la `apache-client` bibliothèque au moment de la compilation.

Reportez-vous à l'exemple de `pom.xml` fichier Maven suivant pour configurer le `ApacheHttpClient`.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.17.290</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId>
  </dependency>
  <!-- By adding the apache-client dependency, ApacheHttpClient will be added to
       the compile classpath so you can configure it. -->
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>apache-client</artifactId>
  </dependency>
</dependencies>
```

Utilisez et configurez `ApacheHttpClient`

Vous pouvez configurer une instance `ApacheHttpClient` tout en créant un client de service, ou vous pouvez configurer une instance unique à partager entre plusieurs clients de service.

Quelle que soit l'approche, vous utilisez le [`ApacheHttpClient.Builder`](#) pour configurer les propriétés du client HTTP basé sur Apache.

Meilleure pratique : dédier une **ApacheHttpClient** instance à un client de service

Si vous devez configurer une instance de `ApacheHttpClient`, nous vous recommandons de créer l'instance dédiée. Vous pouvez le faire en utilisant la `httpClientBuilder` méthode du générateur du client de service. Ainsi, le cycle de vie du client HTTP est géré par le SDK, ce qui permet d'éviter d'éventuelles fuites de mémoire si l'instance n'est pas fermée alors qu'elle n'est plus nécessaire.

L'exemple suivant crée `S3Client` et configure l'instance intégrée de `ApacheHttpClient` with `maxConnections` et `connectionTimeout` values. L'instance HTTP est créée à l'aide de la `httpClientBuilder` méthode de `S3Client.Builder`.

Importations

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Code

```
S3Client s3Client = S3Client // Singleton: Use the s3Client for all requests.
    .builder()
    .httpClientBuilder(ApacheHttpClient.builder()
        .maxConnections(100)
        .connectionTimeout(Duration.ofSeconds(5))
    ).build();

// Perform work with the s3Client.

s3Client.close(); // Requests completed: Close all service clients.
```

Approche alternative : partager une **ApacheHttpClient** instance

Pour réduire l'utilisation des ressources et de la mémoire de votre application, vous pouvez configurer un `ApacheHttpClient` et le partager entre plusieurs clients de service. Le pool de connexions HTTP sera partagé, ce qui réduit l'utilisation des ressources.

Note

Lorsqu'une `ApacheHttpClient` instance est partagée, vous devez la fermer lorsqu'elle est prête à être supprimée. Le SDK ne ferme pas l'instance lorsque le client de service est fermé.

L'exemple suivant configure un client HTTP basé sur Apache qui est utilisé par deux clients de service. L'`ApacheHttpClient` instance configurée est transmise à la `httpClient` méthode de chaque générateur. Lorsque les clients du service et le client HTTP ne sont plus nécessaires, le code les ferme explicitement. Le code ferme le client HTTP en dernier.

Importations

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

Code

```
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .maxConnections(100).build();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
S3Client s3Client =
    S3Client.builder()
        .httpClient(apacheHttpClient).build();

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
        .httpClient(apacheHttpClient).build();

// Perform work with the s3Client and dynamoDbClient.

// Requests completed: Close all service clients.
s3Client.close();
dynamoDbClient.close();
apacheHttpClient.close(); // Explicitly close apacheHttpClient.
```

Exemple de configuration de proxy

L'extrait de code suivant utilise le [générateur de configuration du proxy pour le client HTTP Apache](#).

```
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://example.com:1234"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .addNonProxyHost("host.example.com")
        .build())
    .build();
```

Les propriétés système Java équivalentes pour la configuration du proxy sont indiquées dans l'extrait de ligne de commande suivant.

```
$ java -Dhttp.proxyHost=example.com -Dhttp.proxyPort=1234 -Dhttp.proxyUser=username \
-Dhttp.proxyPassword=password -Dhttp.nonProxyHosts=localhost|host.example.com -cp ...
App
```

La configuration équivalente qui utilise des variables d'environnement est la suivante :

```
// Set the following environment variables.
// $ export HTTP_PROXY="http://username:password@example.com:1234"
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App
```

Note

Le client HTTP Apache ne prend actuellement pas en charge les propriétés du système proxy HTTPS ni la variable d'environnement HTTPS_PROXY.

Configuration du client HTTP basé sur une connexion URL

AWS SDK for Java 2.x II offre un client [UrlConnectionHttpClient](#) HTTP plus léger par rapport au client par défaut. `ApacheHttpClient` `UrlConnectionHttpClient` est basé sur celui de Java [URLConnection](#).

Il se `UrlConnectionHttpClient` charge plus rapidement que le client HTTP basé sur Apache, mais possède moins de fonctionnalités. Comme il se charge plus rapidement, c'est une [bonne solution](#) pour les AWS Lambda fonctions Java.

`UrlConnectionHttpClient` dispose de plusieurs [options configurables](#) auxquelles vous pouvez accéder.

Note

`UrlConnectionHttpClient` ne prend pas en charge la méthode HTTP PATCH. Quelques opérations d' AWS API nécessitent des requêtes PATCH. Ces noms d'opérations commencent généralement par `Update*`. Voici quelques exemples.

- [Plusieurs Update* opérations](#) dans l' AWS Security Hub API et également l'[BatchUpdateFindings](#) opération
- Toutes les [Update*opérations](#) d'API Amazon API Gateway
- [Plusieurs Update* opérations](#) dans l' WorkDocsAPI Amazon

Si vous pouvez utiliser le `UrlConnectionHttpClient`, reportez-vous d'abord à la référence de l'API Service AWS que vous utilisez. Vérifiez si les opérations dont vous avez besoin utilisent l'opération PATCH.

Accédez au `UrlConnectionHttpClient`

Pour configurer et utiliser le `UrlConnectionHttpClient`, vous devez déclarer une dépendance à l'artefact `url-connection-client` Maven dans votre `pom.xml` fichier.

Contrairement au `ApacheHttpClient`, le `UrlConnectionHttpClient` n'est pas automatiquement ajouté à votre projet. L'utilisateur doit donc le déclarer spécifiquement.

L'exemple de `pom.xml` fichier suivant montre les dépendances requises pour utiliser et configurer le client HTTP.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.17.290</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<!-- other dependencies such as s3 or dynamodb -->

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>url-connection-client</artifactId>
  </dependency>
</dependencies>
```

Utilisez et configurez le `URLConnectionHttpClient`

Vous pouvez configurer une instance `URLConnectionHttpClient` tout en créant un client de service, ou vous pouvez configurer une instance unique à partager entre plusieurs clients de service.

Quelle que soit l'approche, vous utilisez le [URLConnectionHttpClient.Builder](#) pour configurer les propriétés du client HTTP basé sur `URLConnection`.

Meilleure pratique : dédier une `URLConnectionHttpClient` instance à un client de service

Si vous devez configurer une instance de `URLConnectionHttpClient`, nous vous recommandons de créer l'`URLConnectionHttpClient` instance dédiée. Vous pouvez le faire en utilisant la `httpClientBuilder` méthode du générateur du client de service. Ainsi, le cycle de vie du client HTTP est géré par le SDK, ce qui permet d'éviter d'éventuelles fuites de mémoire si l'`URLConnectionHttpClient` instance n'est pas fermée alors qu'elle n'est plus nécessaire.

L'exemple suivant crée `S3Client` et configure l'instance intégrée de `URLConnectionHttpClient` with `socketTimeout` et `proxyConfiguration` values. La `proxyConfiguration` méthode utilise une expression Java lambda de type `Consumer<ProxyConfiguration.Builder>`.

Importations

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient;
import java.net.URI;
import java.time.Duration;
```

Code

```
// Singleton: Use the s3Client for all requests.
S3Client s3Client =
    S3Client.builder()
        .httpClientBuilder(UrlConnectionHttpClient.builder()
            .socketTimeout(Duration.ofMinutes(5))
            .proxyConfiguration(proxy -> proxy.endpoint(URI.create("http://
proxy.mydomain.net:8888"))))
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

// Perform work with the s3Client.

s3Client.close(); // Requests completed: Close the s3client.
```

Approche alternative : partager une **UrlConnectionHttpClient** instance

Pour réduire l'utilisation des ressources et de la mémoire de votre application, vous pouvez configurer un `UrlConnectionHttpClient` et le partager entre plusieurs clients de service. Le pool de connexions HTTP sera partagé, ce qui réduit l'utilisation des ressources.

Note

Lorsqu'une `UrlConnectionHttpClient` instance est partagée, vous devez la fermer lorsqu'elle est prête à être supprimée. Le SDK ne ferme pas l'instance lorsque le client de service est fermé.

L'exemple suivant configure un client HTTP basé sur `URLConnection` qui est utilisé par deux clients de service. L'`UrlConnectionHttpClient` instance configurée est transmise à la `httpClient` méthode de chaque générateur. Lorsque les clients du service et le client HTTP ne sont plus nécessaires, le code les ferme explicitement. Le code ferme le client HTTP en dernier.

Imports

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.urlconnection.ProxyConfiguration;
import software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.net.URI;
import java.time.Duration;
```

Code

```
SdkHttpClient urlHttpClient = UrlConnectionHttpClient.create();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
S3Client s3Client =
    S3Client.builder()
        .httpClient(urlHttpClient)
        .defaultsMode(DefaultsMode.IN_REGION)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
        .httpClient(urlHttpClient)
        .defaultsMode(DefaultsMode.IN_REGION)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

// Perform work with the s3Client and dynamoDbClient.

// Requests completed: Close all service clients.
s3Client.close();
dynamoDbClient.close();
urlHttpClient.close();
```

Utiliser **URLConnectionHttpClient** et **ApacheHttpClient** ensemble

Lorsque vous utilisez le `URLConnectionHttpClient` dans votre application, vous devez fournir à chaque client de service une `URLConnectionHttpClient` instance ou une `ApacheHttpClient` instance en utilisant la `httpClientBuilder` méthode du générateur de clients de service.

Une exception se produit si votre programme utilise plusieurs clients de service et que les deux conditions suivantes sont vraies :

- Un client de service est configuré pour utiliser une `URLConnectionHttpClient` instance
- Un autre client de service utilise la valeur par défaut `ApacheHttpClient` sans la créer explicitement avec les `httpClientBuilder()` méthodes `httpClient()` or

L'exception indiquera que plusieurs implémentations HTTP ont été trouvées sur le chemin de classe.

L'exemple d'extrait de code suivant génère une exception.

```
// The dynamoDbClient uses the UrlConnectionHttpClient
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(URLConnectionHttpClient.create())
    .build();

// The s3Client below uses the ApacheHttpClient at runtime, without specifying it.
// An SdkClientException is thrown with the message that multiple HTTP implementations
// were found on the classpath.
S3Client s3Client = S3Client.create();

// Perform work with the s3Client and dynamoDbClient.

dynamoDbClient.close();
s3Client.close();
```

Évitez l'exception en configurant explicitement le `S3Client` avec un `ApacheHttpClient`.

```
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(URLConnectionHttpClient.create())
    .build();

S3Client s3Client = S3Client.builder()
    .httpClient(ApacheHttpClient.create()) // Explicitly build the
    ApacheHttpClient.
    .build();

// Perform work with the s3Client and dynamoDbClient.

dynamoDbClient.close();
s3Client.close();
```

Note

Pour créer explicitement le `ApacheHttpClient`, vous devez [ajouter une dépendance](#) à l'artefact `apache-client` dans votre fichier de projet Maven.

Exemple de configuration de proxy

L'extrait de code suivant utilise le [générateur de configuration du proxy pour le client HTTP de connexion URL](#).

```
SdkHttpClient urlHttpClient = UrlConnectionHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://example.com:1234"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .addNonProxyHost("host.example.com")
        .build())
    .build();
```

Les propriétés système Java équivalentes pour la configuration du proxy sont indiquées dans l'extrait de ligne de commande suivant.

```
$ java -Dhttp.proxyHost=example.com -Dhttp.proxyPort=1234 -Dhttp.proxyUser=username \
-Dhttp.proxyPassword=password -Dhttp.nonProxyHosts=localhost|host.example.com -cp ...
App
```

La configuration équivalente qui utilise des variables d'environnement est la suivante :

```
// Set the following environment variables.
// $ export HTTP_PROXY="http://username:password@example.com:1234"
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkHttpClient apacheHttpClient = UrlConnectionHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();
```

```
// Run the application.  
// $ java -cp ... App
```

Note

Le client HTTP basé sur `URLConnection` ne prend actuellement pas en charge les propriétés du système proxy HTTPS ni la variable d'environnement `HTTPS_PROXY`.

Configuration du client HTTP basé sur Netty

Le client HTTP par défaut pour les opérations asynchrones dans le AWS SDK for Java 2.x est basé sur Netty. [NettyNioAsyncHttpClient](#) Le client basé sur Netty est basé sur le framework réseau asynchrone piloté par les événements du projet Netty.

En tant que client HTTP alternatif, vous pouvez utiliser le nouveau client [HTTP AWS basé sur CRT](#). Cette rubrique explique comment configurer le `NettyNioAsyncHttpClient`.

Accédez au `NettyNioAsyncHttpClient`

Dans la plupart des cas, vous utilisez le `NettyNioAsyncHttpClient` sans configuration explicite dans les programmes asynchrones. Vous déclarez vos clients de service asynchrones et le SDK les configurera `NettyNioAsyncHttpClient` avec des valeurs standard pour vous.

Si vous souhaitez le configurer explicitement `NettyNioAsyncHttpClient` ou l'utiliser avec plusieurs clients de service, vous devez le rendre disponible pour la configuration.

Aucune configuration nécessaire

Lorsque vous déclarez une dépendance à l'égard d'un client de service dans Maven, le SDK ajoute une dépendance d'exécution à `netty-nio-client`artefact. Cela rend la `NettyNioAsyncHttpClient` classe disponible pour votre code au moment de l'exécution, mais pas au moment de la compilation. Si vous ne configurez pas le client HTTP basé sur Netty, vous n'avez pas besoin de spécifier de dépendance pour celui-ci.

Dans l'extrait XML suivant d'un `pom.xml` fichier Maven, la dépendance déclarée avec `<artifactId>dynamodb-enhanced</artifactId>` manière transitive introduit le client HTTP basé sur Netty. Il n'est pas nécessaire de déclarer une dépendance spécifiquement pour celle-ci.

```
<dependencyManagement>
```

```
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>bom</artifactId>
    <version>2.17.290</version>
    <type>pom</type>
    <scope>import</scope>
  </dependency>
</dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb-enhanced</artifactId>
  </dependency>
</dependencies>
```

Avec ces dépendances, vous ne pouvez pas modifier la configuration HTTP, car la `NettyNioAsyncHttpClient` bibliothèque se trouve uniquement sur le chemin de classe d'exécution.

Configuration requise

Pour configurer le `NettyNioAsyncHttpClient`, vous devez ajouter une dépendance à l'`netty-nio-client` artefact au moment de la compilation.

Reportez-vous à l'exemple de `pom.xml` fichier Maven suivant pour configurer le `NettyNioAsyncHttpClient`.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.17.290</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
```



```
        <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
    <!-- By adding the netty-nio-client dependency, NettyNioAsyncHttpClient will
be
        added to the compile classpath so you can configure it. -->
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>netty-nio-client</artifactId>
    </dependency>
</dependencies>
```

Utilisez et configurez le **NettyNioAsyncHttpClient**

Vous pouvez configurer une instance `NettyNioAsyncHttpClient` tout en créant un client de service, ou vous pouvez configurer une instance unique à partager entre plusieurs clients de service.

Quelle que soit l'approche, vous utilisez le [NettyNioAsyncHttpClient.Builder](#) pour configurer les propriétés de l'instance client HTTP basée sur Netty.

Meilleure pratique : dédier une **NettyNioAsyncHttpClient** instance à un client de service

Si vous devez configurer une instance de `NettyNioAsyncHttpClient`, nous vous recommandons de créer une `NettyNioAsyncHttpClient` instance dédiée. Vous pouvez le faire en utilisant la `httpClientBuilder` méthode du générateur du client de service. Ainsi, le cycle de vie du client HTTP est géré par le SDK, ce qui permet d'éviter d'éventuelles fuites de mémoire si l'`NettyNioAsyncHttpClient` instance n'est pas fermée alors qu'elle n'est plus nécessaire.

L'exemple suivant crée une `DynamoDbAsyncClient` instance qui est utilisée par une `DynamoDbEnhancedAsyncClient` instance. L'`DynamoDbAsyncClient` instance contient l'`NettyNioAsyncHttpClient` instance avec des `maxConcurrency` valeurs `connectionTimeout` et. L'instance HTTP est créée à l'aide de la `httpClientBuilder` méthode de `DynamoDbAsyncClient.Builder`.

Importations

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedAsyncClient;
import
    software.amazon.awssdk.enhanced.dynamodb.extensions.AutoGeneratedTimestampRecordExtension;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.time.Duration;
```

Code

```
// DynamoDbAsyncClient is the lower-level client used by the enhanced client.
DynamoDbAsyncClient dynamoDbAsyncClient =
    DynamoDbAsyncClient
        .builder()
            .httpClientBuilder(NettyNioAsyncHttpClient.builder()
                .connectionTimeout(Duration.ofMillis(5_000))
                .maxConcurrency(100)
                .tlsNegotiationTimeout(Duration.ofMillis(3_500)))
            .defaultsMode(DefaultsMode.IN_REGION)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

// Singleton: Use dynamoDbAsyncClient and enhancedClient for all requests.
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient
        .builder()
            .dynamoDbClient(dynamoDbAsyncClient)
            .extensions(AutoGeneratedTimestampRecordExtension.create())
            .build();

// Perform work with the dynamoDbAsyncClient and enhancedClient.

// Requests completed: Close dynamoDbAsyncClient.
dynamoDbAsyncClient.close();
```

Approche alternative : partager une **NettyNioAsyncHttpClient** instance

Pour réduire l'utilisation des ressources et de la mémoire de votre application, vous pouvez en configurer un `NettyNioAsyncHttpClient` et le partager entre plusieurs clients de service. Le pool de connexions HTTP sera partagé, ce qui réduit l'utilisation des ressources.

Note

Lorsqu'une `NettyNioAsyncHttpClient` instance est partagée, vous devez la fermer lorsqu'elle est prête à être supprimée. Le SDK ne ferme pas l'instance lorsque le client de service est fermé.

L'exemple suivant configure un client HTTP basé sur Netty qui est utilisé par deux clients de service. L'`NettyNioAsyncHttpClient` instance configurée est transmise à la `httpClient` méthode de chaque générateur. Lorsque les clients du service et le client HTTP ne sont plus nécessaires, le code les ferme explicitement. Le code ferme le client HTTP en dernier.

Importations

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

Code

```
// Create a NettyNioAsyncHttpClient shared instance.
SdkAsyncHttpClient nettyHttpClient =
    NettyNioAsyncHttpClient.builder().maxConcurrency(100).build();

// Singletons: Use the s3AsyncClient, dbAsyncClient, and enhancedAsyncClient for all
// requests.
S3AsyncClient s3AsyncClient =
    S3AsyncClient.builder()
        .httpClient(nettyHttpClient)
        .build();

DynamoDbAsyncClient dbAsyncClient =
    DynamoDbAsyncClient.builder()
        .httpClient(nettyHttpClient)
        .defaultsMode(DefaultsMode.IN_REGION)

        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

DynamoDbEnhancedAsyncClient enhancedAsyncClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dbAsyncClient)

        .extensions(AutoGeneratedTimestampRecordExtension.create())
        .build();

// Perform work with s3AsyncClient, dbAsyncClient, and enhancedAsyncClient.

// Requests completed: Close all service clients.
```

```
s3AsyncClient.close();
dbAsyncClient.close()
nettyHttpClient.close(); // Explicitly close nettyHttpClient.
```

Exemple de configuration de proxy

L'extrait de code suivant utilise le [générateur de configuration du proxy pour le client HTTP Netty](#).

```
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .nonProxyHosts(Set.of("localhost", "host.example.com"))
        .build())
    .build();
```

Les propriétés système Java équivalentes pour la configuration du proxy sont indiquées dans l'extrait de ligne de commande suivant.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \
-Dhttps.proxyPassword=password -Dhttp.nonProxyHosts=localhost|host.example.com -cp ...
App
```

Important

Pour utiliser l'une des propriétés du système proxy HTTPS, la `scheme` propriété doit être définie dans le code sur `https`. Si la propriété du schéma n'est pas définie dans le code, le schéma est défini par défaut sur HTTP et le SDK recherche uniquement les propriétés `http.*` du système.

La configuration équivalente qui utilise des variables d'environnement est la suivante :

```
// Set the following environment variables.
// $ export HTTPS_PROXY="https://username:password@myproxy:1234"
// $ export NO_PROXY="localhost|host.example.com"
```

```
// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App
```

Configuration de clients AWS HTTP basés sur CRT

Les clients HTTP AWS basés sur le CRT incluent les clients synchrones et asynchrones.

[AwsCrthttpClientAwsCrthttpClient](#) Les clients HTTP AWS basés sur CRT offrent les avantages suivants :

- Temps de démarrage du SDK plus rapide
- Empreinte mémoire réduite
- Temps de latence réduit
- Gestion de l'état de la connexion
- Équilibrage de charge DNS

AWS Composants du SDK basés sur le CRT

Les clients HTTP AWS CRT, décrits dans cette rubrique, et le client S3 AWS basé sur CRT sont des composants différents du SDK.

Les clients HTTP synchrones et asynchrones AWS basés sur le CRT sont des interfaces client HTTP du SDK d'implémentation et sont utilisés pour les communications HTTP générales. Ils constituent des alternatives aux autres clients HTTP synchrones ou asynchrones du SDK avec des avantages supplémentaires.

Le [client S3 AWS basé sur CRT](#) est une implémentation de l'AsyncClientinterface [S3](#) et est utilisé pour travailler avec le service Amazon S3. Il s'agit d'une alternative à l'implémentation Java de l'S3AsyncClientinterface et offre plusieurs avantages.

Bien que les deux composants utilisent des bibliothèques issues du [AWS Common Runtime](#), les clients HTTP AWS basés sur CRT n'utilisent pas la [bibliothèque aws-c-s 3](#) et ne prennent pas en charge les fonctionnalités de l'API de [téléchargement partitionné S3](#). Le client S3 AWS basé sur

CRT, en revanche, a été spécialement conçu pour prendre en charge les fonctionnalités de l'API de téléchargement partitionné S3.

Accédez aux clients HTTP AWS basés sur CRT

Avant de pouvoir utiliser les clients HTTP AWS CRT, ajoutez l'`aws-crt-client` artefact avec une version minimale de 2.22.0 aux dépendances de votre projet.

Le Maven suivant `pom.xml` montre les clients HTTP AWS basés sur le CRT déclarés à l'aide du mécanisme de nomenclature (BOM).

```
<project>
  <properties>
    <aws.sdk.version>2.22.0</aws.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
    </dependency>
  </dependencies>
</project>
```

Visitez le dépôt central de Maven pour obtenir la [dernière version](#).

Utiliser et configurer un client AWS HTTP CRT

Vous pouvez configurer un client HTTP AWS CRT tout en créant un client de service, ou vous pouvez configurer une instance unique à partager entre plusieurs clients de service.

Quelle que soit l'approche, vous utilisez un générateur pour [configurer les propriétés](#) de l'instance client HTTP AWS basée sur CRT.

Meilleure pratique : dédier une instance à un client de service

Si vous devez configurer une instance d'un client HTTP AWS CRT, nous vous recommandons de la dédier en la créant avec le client de service. Vous pouvez le faire en utilisant la `httpClientBuilder` méthode du générateur du client de service. Ainsi, le cycle de vie du client HTTP est géré par le SDK, ce qui permet d'éviter d'éventuelles fuites de mémoire si l'instance du client HTTP AWS basée sur CRT n'est pas fermée alors qu'elle n'est plus nécessaire.

L'exemple suivant crée un client de service S3 et configure un client HTTP AWS CRT avec `connectionTimeout` des valeurs et `maxConcurrency`

Synchronous client

Importations

```
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;  
import software.amazon.awssdk.services.s3.S3Client;  
import java.time.Duration;
```

Code

```
// Singleton: Use s3Client for all requests.  
S3Client s3Client = S3Client.builder()  
    .httpClientBuilder(AwsCrtHttpClient  
        .builder()  
        .connectionTimeout(Duration.ofSeconds(3))  
        .maxConcurrency(100))  
    .build();  
  
// Perform work with the s3Client.  
  
// Requests completed: Close the s3Client.  
s3Client.close();
```

Asynchronous client

Importations

```
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;  
import software.amazon.awssdk.services.s3.S3AsyncClient;  
import java.time.Duration;
```

Code

```
// Singleton: Use s3AsyncClient for all requests.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .httpClientBuilder(AwsCrtAsyncHttpClient
        .builder()
        .connectionTimeout(Duration.ofSeconds(3))
        .maxConcurrency(100))
    .build();

// Perform work with the s3AsyncClient.

// Requests completed: Close the s3AsyncClient.
s3AsyncClient.close();
```

Approche alternative : partager une instance

Pour réduire l'utilisation des ressources et de la mémoire de votre application, vous pouvez configurer un client HTTP AWS CRT et le partager entre plusieurs clients de service. Le pool de connexions HTTP sera partagé, ce qui réduit l'utilisation des ressources.

Note

Lorsqu'une instance client AWS HTTP CRT est partagée, vous devez la fermer lorsqu'elle est prête à être supprimée. Le SDK ne ferme pas l'instance lorsque le client de service est fermé.

L'exemple suivant configure une instance de client HTTP AWS CRT avec `connectionTimeout` des valeurs et `maxConcurrency`. L'instance configurée est transmise à la `httpClient` méthode du générateur de chaque client de service. Lorsque les clients du service et le client HTTP ne sont plus nécessaires, ils sont explicitement fermés. Le client HTTP est fermé en dernier.

Synchronous client

Importations

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;
```



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Code

```
// Create an AwsCrtHttpClient shared instance.
SdkHttpClient crtHttpClient = AwsCrtHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(3))
    .maxConcurrency(100)
    .build();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
S3Client s3Client = S3Client.builder()
    .httpClient(crtHttpClient)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.crea
    .defaultsMode(DefaultsMode.IN_REGION)
    .region(Region.US_EAST_1)
    .build();

DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(crtHttpClient)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.crea
    .defaultsMode(DefaultsMode.IN_REGION)
    .region(Region.US_EAST_1)
    .build();

// Requests completed: Close all service clients.
s3Client.close();
dynamoDbClient.close();
crtHttpClient.close(); // Explicitly close crtHttpClient.
```

Asynchronous client

Importations

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import java.time.Duration;
```

Code

```
// Create an AwsCrtAsyncHttpClient shared instance.
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(3))
    .maxConcurrency(100)
    .build();

// Singletons: Use the s3AsyncClient and dynamoDbAsyncClient for all requests.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .httpClient(crtAsyncHttpClient)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .defaultsMode(DefaultsMode.IN_REGION)
    .region(Region.US_EAST_1)
    .build();

DynamoDbAsyncClient dynamoDbAsyncClient = DynamoDbAsyncClient.builder()
    .httpClient(crtAsyncHttpClient)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .defaultsMode(DefaultsMode.IN_REGION)
    .region(Region.US_EAST_1)
    .build();

// Requests completed: Close all service clients.
s3AsyncClient.close();
dynamoDbAsyncClient.close();
crtAsyncHttpClient.close(); // Explicitly close crtAsyncHttpClient.
```

Définir un client HTTP AWS CRT comme client par défaut

Vous pouvez configurer votre fichier de compilation Maven pour que le SDK utilise un client HTTP AWS CRT comme client HTTP par défaut pour les clients de service.

Pour ce faire, ajoutez un `exclusions` élément avec les dépendances du client HTTP par défaut à chaque artefact du client de service.

Dans l'exemple suivant, le SDK utilise un client HTTP AWS CRT pour les services S3. Si le client de service de votre code est un `S3AsyncClient`, le SDK utilise `AwsCrtAsyncHttpClient`.

le client de service est un client S3, le SDK utilise `AwsCrtHttpClient`. Avec cette configuration, le client HTTP asynchrone par défaut basé sur Netty et le HTTP synchrone basé sur Apache par défaut ne sont pas disponibles.

```
<project>
  <properties>
    <aws.sdk.version>VERSION</aws.sdk.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
      <version>${aws.sdk.version}</version>
      <exclusions>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>netty-nio-client</artifactId>
        </exclusion>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>apache-client</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
    </dependency>
  </dependencies>
</project>
```

Visitez le dépôt central de Maven pour obtenir la dernière valeur de [VERSION](#).

Note

Si plusieurs clients de service sont déclarés dans un `pom.xml` fichier, ils ont tous besoin de l'élément `exclusions` XML.

Utiliser une propriété système Java

Pour utiliser les clients HTTP AWS CRT comme protocole HTTP par défaut pour votre application, vous pouvez définir la propriété système Java sur une valeur `software.amazon.awssdk.http.async.service.impl` de `software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient`.

Pour définir lors du démarrage de l'application, exécutez une commande similaire à la suivante.

```
java app.jar -Dsoftware.amazon.awssdk.http.async.service.impl=\
software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient
```

Utilisez l'extrait de code suivant pour définir la propriété système dans le code de votre application.

```
System.setProperty("software.amazon.awssdk.http.async.service.impl",
"software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient");
```

Note

Vous devez ajouter une dépendance à `aws-crt-client` de votre `pom.xml` fichier lorsque vous utilisez une propriété système pour configurer l'utilisation des clients HTTP AWS CRT.

Configuration avancée des clients HTTP AWS basés sur CRT

Vous pouvez utiliser différents paramètres de configuration des clients HTTP AWS basés sur CRT, notamment la configuration de l'état de la connexion et le temps d'inactivité maximal. Vous pouvez consulter les [options de configuration disponibles](#) pour `AwsCrtAsyncHttpClient`. Vous pouvez configurer les mêmes options pour `AwsCrtHttpClient`.

Configuration de l'état de la connexion

Vous pouvez configurer la configuration de l'état de la connexion pour les clients HTTP AWS basés sur CRT en utilisant la `connectionHealthConfiguration` méthode du générateur de clients HTTP.

L'exemple suivant crée un client de service S3 qui utilise une instance de client HTTP AWS CRT configurée avec une configuration d'état de connexion et une durée d'inactivité maximale pour les connexions.

Synchronous client

Importations

```
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Code

```
// Singleton: Use the s3Client for all requests.
S3Client s3Client = S3Client.builder()
    .httpClientBuilder(AwsCrtHttpClient
        .builder()
        .connectionHealthConfiguration(builder -> builder
            .minimumThroughputInBps(32000L)
            .minimumThroughputTimeout(Duration.ofSeconds(3)))
        .connectionMaxIdleTime(Duration.ofSeconds(5)))
    .build();

// Perform work with s3Client.

// Requests complete: Close the service client.
s3Client.close();
```

Asynchronous client

Importations

```
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import java.time.Duration;
```

Code

```
// Singleton: Use the s3AsyncClient for all requests.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .httpClientBuilder(AwsCrtAsyncHttpClient
        .builder()
        .connectionHealthConfiguration(builder -> builder
            .minimumThroughputInBps(32000L)
            .minimumThroughputTimeout(Duration.ofSeconds(3)))
        .connectionMaxIdleTime(Duration.ofSeconds(5)))
```

```
.build();

// Perform work with s3AsyncClient.

// Requests complete: Close the service client.
s3AsyncClient.close();
```

Support HTTP/2

Le protocole HTTP/2 n'est pas encore pris en charge dans les clients HTTP AWS basés sur CRT, mais il est prévu pour une future version.

En attendant, si vous utilisez des clients de service qui nécessitent un support HTTP/2 tel que le [KinesisAsyncClient](#) ou le [TranscribeStreamingAsyncClient](#), pensez à utiliser le à la [NettyNioAsyncHttpClient](#) place.

Exemple de configuration de proxy

L'extrait de code suivant montre l'utilisation de [ProxyConfiguration.Builder](#) celui que vous utilisez pour configurer les paramètres du proxy dans le code.

Synchronous client

Importations

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;
import software.amazon.awssdk.http.crt.ProxyConfiguration;
```

Code

```
SdkHttpClient crtHttpClient = AwsCrtHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .nonProxyHosts(Set.of("localhost", "host.example.com")))
    .build()
    .build();
```

Asynchronous client

Importations

```
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;  
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;  
import software.amazon.awssdk.http.crt.ProxyConfiguration;
```

Code

```
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()  
    .proxyConfiguration(ProxyConfiguration.builder()  
        .scheme("https")  
        .host("myproxy")  
        .port(1234)  
        .username("username")  
        .password("password")  
        .nonProxyHosts(Set.of("localhost", "host.example.com"))  
        .build())  
    .build();
```

Les propriétés système Java équivalentes pour la configuration du proxy sont indiquées dans l'extrait de ligne de commande suivant.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \  
-Dhttps.proxyPassword=password -Dhttp.nonProxyHosts=localhost|host.example.com -cp ...  
App
```

Important

Pour utiliser l'une des propriétés du système proxy HTTPS, la `scheme` propriété doit être définie dans le code sur `https`. Si la propriété du schéma n'est pas définie dans le code, le schéma est défini par défaut sur HTTP et le SDK recherche uniquement les propriétés `http.*` du système.

La configuration équivalente qui utilise des variables d'environnement est la suivante :

```
// Set the following environment variables.  
// $ export HTTPS_PROXY="https://username:password@myproxy:1234"
```

```
// $ export NO_PROXY="localhost|host.example.com"

// Set the 'useSystemPropertyValues' to false on the proxy configuration.
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .useSystemPropertyValues(Boolean.FALSE)
        .build())
    .build();

// Run the application.
// $ java -cp ... App
```

Configuration des proxys HTTP

Vous pouvez configurer les proxys HTTP à l'aide de code, en définissant les propriétés du système Java ou en définissant des variables d'environnement.

Configurer dans le code

Vous configurez les proxys dans le code avec un `ProxyConfiguration` générateur spécifique au client lorsque vous créez le client de service. Le code suivant montre un exemple de configuration de proxy pour un client HTTP basé sur Apache utilisé par un client de service Amazon S3.

```
SdkHttpClient httpClient1 = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://proxy.example.com"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .build())
    .build();

S3Client s3Client = S3Client.builder()
    .httpClient(httpClient)
    .build();
```

La section de cette rubrique consacrée à chaque client HTTP présente un exemple de configuration de proxy.

- [Client HTTP Apache](#)
- [Client HTTP basé sur une connexion URL](#)

- [Client HTTP basé sur Netty](#)
- [AWS Client HTTP basé sur CRT](#)

Configuration de proxys HTTP avec des paramètres externes

Même si vous n'utilisez pas explicitement de `ProxyConfiguration` générateur dans le code, le SDK recherche des paramètres externes pour configurer une configuration de proxy par défaut.

Par défaut, le SDK recherche d'abord les propriétés du système JVM. Si une seule propriété est trouvée, le SDK utilise la valeur et toutes les autres valeurs des propriétés du système. Si aucune propriété système n'est disponible, le SDK recherche les variables d'environnement du proxy.

Le SDK peut utiliser les propriétés du système Java et les variables d'environnement suivantes.

Propriétés du système Java

Propriété du système	Description	Support client HTTP
<code>Http.ProxyHost</code>	Nom d'hôte du serveur proxy HTTP	Tous
<code>Http.ProxyPort</code>	Numéro de port du serveur proxy HTTP	Tous
<code>Http.ProxyUser</code>	Nom d'utilisateur pour l'authentification par proxy HTTP	Tous
<code>HTTP.ProxyPassword</code>	Mot de passe pour l'authentification par proxy HTTP	Tous
<code>http.nonProxyHosts</code>	Liste des hôtes à atteindre directement, en contournant le proxy. Cette liste est également valide lorsque le protocole HTTPS est utilisé.	Tous
<code>Https.proxyHost</code>	Nom d'hôte du serveur proxy HTTPS	Netty, CRT

Propriété du système	Description	Support client HTTP
Https.ProxyPort	Numéro de port du serveur proxy HTTPS	Netty, CRT
Https.proxyUser	Nom d'utilisateur pour l'authentification par proxy HTTPS	Netty, CRT
HTTPS.ProxyPassword	Mot de passe pour l'authentification par proxy HTTPS	Netty, CRT

Variables d'environnement

Variable d'environnement	Description	Support client HTTP
PROXY_HTTP ¹	Une URL valide avec un schéma HTTP	Tous
HTTPS_PROXY ¹	Une URL valide avec un schéma HTTPS	Netty, CRT
AUCUN PROXY ²	Liste des hôtes à atteindre directement, en contournant le proxy. La liste est valide pour HTTP et HTTPS.	Tous

Afficher les clés et les notes de bas de page

Tous : tous les clients HTTP proposés par le SDK—`URLConnectionHttpClient`, `ApacheHttpClientNettyNioAsyncHttpClient`, `AwsCrtAsyncHttpClient`.

Netty - Le client HTTP basé sur Netty (). `NettyNioAsyncHttpClient`

CRT - Les clients HTTP AWS basés sur CRT, (et) `AwsCrtHttpClient`. `AwsCrtAsyncHttpClient`

¹ La variable d'environnement demandée, qu'elle soit `HTTP_PROXY` ou non `HTTPS_PROXY`, dépend du paramètre du schéma défini dans celui du client. `ProxyConfiguration` Le schéma par défaut est

HTTP. L'extrait suivant montre comment modifier le schéma en HTTPS utilisé pour la résolution des variables d'environnement.

```
SdkHttpClient httpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .build())
    .build();
```

² La variable d'NO_PROXY environnement prend en charge une combinaison de séparateurs « | » et « , » entre les noms d'hôtes. Les noms d'hôtes peuvent inclure le caractère générique « * ».

Utiliser une combinaison de paramètres

Vous pouvez utiliser une combinaison de paramètres de proxy HTTP dans le code, les propriétés du système et les variables d'environnement.

Exemple — configuration fournie par une propriété du système et par un code

```
// Command line with the proxy password set as a system property.
$ java -Dhttp.proxyPassword=SYS_PROP_password -cp ... App

// Since the 'useSystemPropertyValues' setting is 'true' (the default), the SDK will
// supplement
// the proxy configuration in code with the 'http.proxyPassword' value from the system
// property.
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://localhost:1234"))
        .username("username")
        .build())
    .build();

// Use the apache HTTP client with proxy configuration.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(apacheHttpClient)
    .build();
```

Le SDK résout les paramètres de proxy suivants.

```
Host = localhost
Port = 1234
```

```
Password = SYS_PROP_password
UserName = username
Non ProxyHost = null
```

Exemple — les propriétés du système et les variables d'environnement sont disponibles

Le `ProxyConfiguration` générateur de chaque client HTTP propose des paramètres nommés `useSystemPropertyValues` et `useEnvironmentVariablesValues`. Par défaut, les deux paramètres sont `true`. Lorsque `true` le SDK utilise automatiquement les valeurs des propriétés du système ou des variables d'environnement pour les options qui ne sont pas fournies par le `ProxyConfiguration` générateur.

Important

Les propriétés du système ont priorité sur les variables d'environnement. Si une propriété du système proxy HTTP est détectée, le SDK extrait toutes les valeurs des propriétés du système et aucune des variables d'environnement. Si vous souhaitez hiérarchiser les variables d'environnement par rapport `useSystemPropertyValues` aux propriétés du système, définissez sur `false`.

Dans cet exemple, les paramètres suivants sont disponibles au moment de l'exécution :

```
// System properties
http.proxyHost=SYS_PROP_HOST.com
http.proxyPort=2222
http.password=SYS_PROP_PASSWORD
http.user=SYS_PROP_USER

// Environment variables
HTTP_PROXY="http://EnvironmentUser:EnvironmentPassword@ENV_VAR_HOST:3333"
NO_PROXY="environmentnonproxy.host,environmentnonproxy2.host:1234"
```

Le client de service est créé avec l'une des instructions suivantes. Aucune des instructions ne définit explicitement un paramètre de proxy.

```
DynamoDbClient client = DynamoDbClient.create();
DynamoDbClient client = DynamoDbClient.builder().build();
DynamoDbClient client = DynamoDbClient.builder()
    .httpClient(ApacheHttpClient.builder()
        .proxyConfiguration(ProxyConfiguration.builder()
```

```
        .build())
    .build())
    .build();
```

Les paramètres de proxy suivants sont résolus par le SDK :

```
Host = SYS_PROP_HOST.com
Port = 2222
Password = SYS_PROP_PASSWORD
UserName = SYS_PROP_USER
Non ProxyHost = null
```

Comme le client de service possède des paramètres de proxy par défaut, le SDK recherche les propriétés du système, puis les variables d'environnement. Étant donné que les paramètres des propriétés du système ont priorité sur les variables d'environnement, le SDK utilise uniquement les propriétés du système.

Si l'utilisation des propriétés du système est modifiée `false` comme indiqué dans le code suivant, le SDK résout uniquement les variables d'environnement.

```
DynamoDbClient client = DynamoDbClient.builder()
    .httpClient(ApacheHttpClient.builder()
        .proxyConfiguration(ProxyConfiguration.builder()
            .useSystemPropertyValues(Boolean.FALSE)
            .build())
        .build())
    .build();
```

Les paramètres de proxy résolus à l'aide du protocole HTTP sont les suivants :

```
Host = ENV_VAR_HOST
Port = 3333
Password = EnvironmentPassword
UserName = EnvironmentUser
Non ProxyHost = environmentnonproxy.host, environmentnonproxy2.host:1234
```

Gestion des exceptions pour AWS SDK for Java 2.x

Comprendre comment et quand le kit AWS SDK for Java 2.x lève des exceptions est important pour développer des applications de grande qualité à l'aide du kit 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 ?

Le kit AWS SDK for Java utilise les exceptions du runtime (ou exceptions non contrôlées) à la place des exceptions contrôlées 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.

AwsServiceException (et sous-classes)

[AwsServiceException](#) est l'exception la plus courante que vous rencontrerez lors de l'utilisation du AWS SDK for Java. [AwsServiceException](#) est une sous-classe de la catégorie plus générale [SdkServiceException](#). [AwsServiceException](#) représentent 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, vous Amazon EC2 renverrez une réponse d'erreur et tous les détails de cette réponse d'erreur seront inclus dans le [AwsServiceException](#) message envoyé.

Lorsque vous rencontrez un [AwsServiceException](#), 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.

[AwsServiceException](#) vous fournit des informations telles que :

- Code d'état HTTP retourné
- Code AWS d'erreur renvoyé
- Message d'erreur détaillé du service de la [AwsErrorDetails](#) classe
- AWSID de demande pour la demande qui a échoué

Dans la plupart des cas, une sous-classe spécifique au service `AwsServiceException` est créée pour permettre aux développeurs de contrôler avec précision la gestion des cas d'erreur par le biais de « catch blocks ». La référence de l'API du SDK Java pour [AwsServiceException](#) affiche le grand nombre de `AwsServiceException` sous-classes. Utilisez les liens des sous-classes pour effectuer une analyse détaillée des exceptions générées par un service.

Par exemple, les liens suivants vers la référence de l'API du SDK présentent les hiérarchies d'exceptions pour certaines exceptions courantes. Services AWS La liste des sous-classes figurant sur chaque page indique les exceptions spécifiques que votre code peut intercepter.

- [Amazon S3](#)
- [DynamoDB](#)
- [Amazon SQS](#)

Pour en savoir plus sur une exception, `errorCode` inspectez l'[AwsErrorDetails](#) objet. Vous pouvez utiliser cette `errorCode` valeur pour rechercher des informations dans l'API du guide des services. Par exemple, si un `S3Exception` est détecté et que sa `AwsErrorDetails#errorCode()` valeur est le cas `InvalidRequest`, utilisez la [liste des codes d'erreur](#) dans le manuel Amazon S3 API Reference pour obtenir plus de détails.

SdkClientException

[SdkClientException](#) 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 `SdkClientException` est généralement plus grave qu'un `SdkServiceException` et indique un problème majeur qui empêche le client de faire des appels de service aux AWS services. Par exemple, le kit AWS SDK for Java génère une exception `SdkClientException` si aucune connexion réseau n'est disponible lorsque vous essayez d'appeler une opération sur l'un des clients.

Exceptions et comportement en cas de nouvelle tentative

Le SDK for Java réessaie les demandes relatives à [plusieurs exceptions côté client](#) et aux codes d'état HTTP qu'il reçoit des réponses. Service AWS Ces erreurs sont traitées dans le cadre de l'héritage `RetryMode` que les clients du service utilisent par défaut. La référence de l'API Java pour [RetryMode](#) décrit les différentes manières de configurer le mode.

Pour personnaliser les exceptions et les codes d'état HTTP qui déclenchent des tentatives automatiques, configurez votre client de service avec un [RetryPolicy](#) ajout [RetryOnExceptionsCondition](#) et des [RetryOnStatusCodeCondition](#) instances.

Journalisation avec le SDK pour Java 2.x

Il AWS SDK for Java 2.x utilise [SLF4J](#), qui est 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 le Java Logging Framework et Apache [Log4j 2](#), entre autres. Cette rubrique explique comment utiliser Log4j 2 comme système de journalisation pour travailler avec le SDK.

Fichier de configuration Log4j 2

Vous utilisez généralement un fichier de configuration, nommé `log4j2.xml` Log4j 2. Vous trouverez ci-dessous des exemples de fichiers de configuration. Pour en savoir plus sur les valeurs utilisées dans le fichier de configuration, consultez le [manuel pour la configuration de Log4j](#).

Le `log4j2.xml` fichier doit se trouver sur le chemin de classe au démarrage de l'application. Pour un projet Maven, placez le fichier dans le `<project-dir>/src/main/resources` répertoire.

Le fichier `log4j2.xml` de configuration spécifie des propriétés telles que le [niveau](#) de journalisation, l'endroit où la sortie de journalisation est envoyée (par exemple, [vers un fichier ou vers la console](#)) et le [format de la sortie](#). Le niveau de journalisation indique le niveau de détail généré par Log4j 2. [Log4j 2 prend en charge le concept de hiérarchies de journalisation multiples](#). Le niveau de journalisation est défini de manière indépendante pour chaque hiérarchie. La principale hiérarchie de journalisation que vous utilisez avec le AWS SDK for Java 2.x est `software.amazon.awssdk`.

Ajouter une dépendance à la journalisation

Pour configurer la liaison Log4j 2 pour SLF4J dans votre fichier de compilation, utilisez ce qui suit.

Maven

Ajoutez les éléments suivants à votre `pom.xml` fichier.

```
...
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-slf4j2-impl</artifactId>
```



```
<version>VERSION</version>
</dependency>
...
```

Gradle–Kotlin DSL

Ajoutez ce qui suit à votre `build.gradle.kts` fichier.

```
...
dependencies {
    ...
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl:VERSION")
    ...
}
...
```

2.20.0 À utiliser pour la version minimale de l'`log4j-slf4j2-impl` artefact. Pour la dernière version, utilisez la version publiée sur [Maven Central](#). Remplacez `VERSION` par la version que vous utiliserez.

Erreurs et avertissements spécifiques au SDK

Nous vous recommandons de toujours laisser la hiérarchie des enregistreurs « `software.amazon.awssdk` » définie sur « `WARN` » pour intercepter les messages importants provenant des bibliothèques clientes du SDK. Par exemple, si le client Amazon S3 détecte que votre application n'a pas correctement fermé une application `InputStream` et qu'elle est susceptible de provoquer une fuite de ressources, il 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 `log4j2.xml` fichier suivant définit le paramètre sur `rootLogger` « `WARN` », ce qui entraîne l'affichage de messages d'avertissement et d'erreur provenant de tous les enregistreurs de l'application, y compris ceux de la hiérarchie « `software.amazon.awssdk` ». Vous pouvez également définir explicitement la hiérarchie de l'enregistreur « `software.amazon.awssdk` » sur « `WARN` » si elle est utilisée. `<Root level="ERROR">`

Exemple de fichier de configuration Log4j2.xml

Cette configuration enregistre les messages aux niveaux « `ERROR` » et « `WARN` » sur la console pour toutes les hiérarchies d'enregistreurs.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
    </Console>
  </Appenders>

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender"/>
    </Root>
  </Loggers>
</Configuration>
```

Enregistrement du résumé des demandes/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 [SdkServiceException](#) des objets du SDK en cas d'échec d'un appel de service, et peuvent également être signalés via le niveau de journal « DEBUG » de l'enregistreur « software.amazon.awssdk.request ».

Le `log4j2.xml` fichier suivant permet de résumer les demandes et les réponses.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
    </Console>
  </Appenders>

  <Loggers>
    <Root level="ERROR">
      <AppenderRef ref="ConsoleAppender"/>
    </Root>
    <Logger name="software.amazon.awssdk" level="WARN" />
    <Logger name="software.amazon.awssdk.request" level="DEBUG" />
  </Loggers>
</Configuration>
```

Voici un exemple de la sortie du journal:

```
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
DefaultSdkHttpRequestFullRequest(httpMethod=POST, protocol=https, host=dynamodb.us-
east-1.amazonaws.com, encodedPath=/, headers=[amz-sdk-invocation-id, Content-Length,
Content-Type, User-Agent, X-Amz-Target], queryParameters=[])
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Received
successful response: 200, Request ID:
QS9DUMME2NHEDH8TGT9N5V530JVV4KQNS05AEMVJF66Q9ASUAAJG, Extended Request ID: not
available
```

Si vous êtes intéressé uniquement par l'identifiant de demande, utilisez `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`.

Journalisation du SDK au niveau du débogage

Si vous avez besoin de plus de détails sur le fonctionnement du SDK, vous pouvez définir le niveau de journalisation de `software.amazon.awssdk` enregistreur sur `DEBUG`. À ce niveau, le SDK produit une grande quantité de détails. Nous vous recommandons donc de définir ce niveau pour résoudre les erreurs à l'aide de tests d'intégration.

À ce niveau de journalisation, le SDK enregistre les informations relatives à la configuration, à la résolution des identifiants, aux intercepteurs d'exécution, à l'activité TLS de haut niveau, à la signature des demandes, etc.

Voici un échantillon d'instructions produites par le SDK au `DEBUG` niveau d'un `S3Client#listBuckets()` appel.

```
DEBUG s.a.a.r.p.AwsRegionProviderChain:57 - Unable to load region from
software.amazon.awssdk.regions.providers.SystemSettingsRegionProvider@324dcd31:Unable
to load region from system settings. Region must be specified either via environment
variable (AWS_REGION) or system property (aws.region).
DEBUG s.a.a.c.i.h.l.ClasspathSdkHttpServiceProvider:85 - The HTTP implementation loaded
is software.amazon.awssdk.http.apache.ApacheSdkHttpService@a23a01d
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Creating an interceptor
chain that will apply interceptors in the following order:
[software.amazon.awssdk.core.internal.interceptor.HttpChecksumValidationInterceptor@69b2f8e5,
software.amazon.awssdk.awscore.interceptor.HelpfulUnknownHostExceptionInterceptor@6331250e,
software.amazon.awssdk.awscore.eventstream.EventStreamInitialRequestInterceptor@a10c1b5,
software.amazon.awssdk.awscore.interceptor.TraceIdExecutionInterceptor@644abb8f,
software.amazon.awssdk.services.s3.auth.scheme.internal.S3AuthSchemeInterceptor@1a411233,
software.amazon.awssdk.services.s3.endpoints.internal.S3ResolveEndpointInterceptor@70325d20,
software.amazon.awssdk.services.s3.endpoints.internal.S3RequestSetEndpointInterceptor@7c2327fa,
software.amazon.awssdk.services.s3.internal.handlers.StreamingRequestInterceptor@4d847d32,
```

```
software.amazon.awssdk.services.s3.internal.handlers.CreateBucketInterceptor@5f462e3b,
software.amazon.awssdk.services.s3.internal.handlers.CreateMultipartUploadRequestInterceptor@3
software.amazon.awssdk.services.s3.internal.handlers.DecodeUrlEncodedResponseInterceptor@58065
software.amazon.awssdk.services.s3.internal.handlers.GetBucketPolicyInterceptor@3605c4d3,
software.amazon.awssdk.services.s3.internal.handlers.S3ExpressChecksumInterceptor@585c13de,
software.amazon.awssdk.services.s3.internal.handlers.AsyncChecksumValidationInterceptor@187eb9
software.amazon.awssdk.services.s3.internal.handlers.SyncChecksumValidationInterceptor@726a6b9
software.amazon.awssdk.services.s3.internal.handlers.EnableTrailingChecksumInterceptor@6ad11a5
software.amazon.awssdk.services.s3.internal.handlers.ExceptionTranslationInterceptor@522b2631,
software.amazon.awssdk.services.s3.internal.handlers.GetObjectInterceptor@3ff57625,
software.amazon.awssdk.services.s3.internal.handlers.CopySourceInterceptor@1ee29c84,
software.amazon.awssdk.services.s3.internal.handlers.ObjectMetadataInterceptor@7c8326a4]
DEBUG s.a.a.u.c.CachedSupplier:85 - (SsoOidcTokenProvider()) Cached value is stale and
will be refreshed.
...
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Creating an interceptor
chain that will apply interceptors in the following order:
[software.amazon.awssdk.core.internal.interceptor.HttpChecksumValidationInterceptor@51351f28,
software.amazon.awssdk.awscore.interceptor.HelpfulUnknownHostExceptionInterceptor@21618fa7,
software.amazon.awssdk.awscore.eventstream.EventStreamInitialRequestInterceptor@15f2eda3,
software.amazon.awssdk.awscore.interceptor.TraceIdExecutionInterceptor@34cf294c,
software.amazon.awssdk.services.sso.auth.scheme.internal.SsoAuthSchemeInterceptor@4d7aaca2,
software.amazon.awssdk.services.sso.endpoints.internal.SsoResolveEndpointInterceptor@604b1e1d,
software.amazon.awssdk.services.sso.endpoints.internal.SsoRequestSetEndpointInterceptor@625668
...
DEBUG s.a.a.request:85 - Sending Request: DefaultSdkHttpFullRequest(httpMethod=GET,
protocol=https, host=portal.sso.us-east-1.amazonaws.com, encodedPath=/federation/
credentials, headers=[amz-sdk-invocation-id, User-Agent, x-amz-sso_bearer_token],
queryParameters=[role_name, account_id])
DEBUG s.a.a.c.i.h.p.s.SigningStage:85 - Using SelectedAuthScheme: smithy.api#noAuth
DEBUG s.a.a.h.a.i.c.SdkTlsSocketFactory:366 - Connecting socket to portal.sso.us-
east-1.amazonaws.com/18.235.195.183:443 with timeout 2000
...
DEBUG s.a.a.requestId:85 - Received successful response: 200, Request ID: bb4f40f4-
e920-4b5c-8648-58f26e7e08cd, Extended Request ID: not available
DEBUG s.a.a.request:85 - Received successful response: 200, Request ID: bb4f40f4-
e920-4b5c-8648-58f26e7e08cd, Extended Request ID: not available
DEBUG s.a.a.u.c.CachedSupplier:85 -
  (software.amazon.awssdk.services.sso.auth.SsoCredentialsProvider@b965857) Successfully
  refreshed cached value. Next Prefetch Time: 2024-04-25T22:03:10.097Z. Next Stale Time:
  2024-04-25T22:05:30Z
DEBUG s.a.a.c.i.ExecutionInterceptorChain:85 - Interceptor
'software.amazon.awssdk.services.s3.endpoints.internal.S3RequestSetEndpointInterceptor@7c2327f
modified the message with its modifyHttpRequest method.
```

```

...
DEBUG s.a.a.c.i.h.p.s.SigningStage:85 - Using SelectedAuthScheme: aws.auth#sigv4
...
DEBUG s.a.a.a.s.Aws4Signer:85 - AWS4 Canonical Request: GET
...
DEBUG s.a.a.h.a.a.i.s.DefaultV4RequestSigner:85 - AWS4 String to sign: AWS4-HMAC-SHA256
20240425T210631Z
20240425/us-east-1/s3/aws4_request
aafb7784627fa7a49584256cb746279751c48c2076f813259ef767ecce304d64
DEBUG s.a.a.h.a.i.c.SdkTlsSocketFactory:366 - Connecting socket to s3.us-
east-1.amazonaws.com/52.217.41.86:443 with timeout 2000
...

```

Le `log4j2.xml` fichier suivant configure la sortie précédente.

```

<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%-5p %c{1.}:%L - %m%n" />
    </Console>
  </Appenders>

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender"/>
    </Root>
    <Logger name="software.amazon.awssdk" level="DEBUG" />
  </Loggers>
</Configuration>

```

Activer l'enregistrement des câbles

Il peut être utile de voir les demandes et réponses exactes que le SDK for Java 2.x envoie et reçoit. Si vous avez besoin d'accéder à ces informations, vous pouvez les activer temporairement en ajoutant la configuration nécessaire en fonction du client HTTP utilisé par le client de service.

Par défaut, les clients de service synchrones, tels que le [S3Client](#), utilisent un Apache sous-jacent `HttpClient`, et les clients de service asynchrones, tels que le [S3 AsyncClient](#), utilisent un client HTTP non bloquant `Netty`.

Voici une liste des clients HTTP que vous pouvez utiliser pour les deux catégories de clients de service :

Clients HTTP synchrones	Clients HTTP asynchrones
ApacheHttpClient (par défaut)	NettyNioAsyncHttpClient (par défaut)
URLConnectionHttpClient	AwsCrtAsyncHttpClient

Consultez l'onglet approprié ci-dessous pour connaître les paramètres de configuration que vous devez ajouter en fonction du client HTTP sous-jacent.

Warning

Nous vous recommandons d'utiliser uniquement la journalisation du réseau filaire à des fins de débogage. Désactivez-la dans vos environnements de production, car elle peut consigner des données sensibles. Elle enregistre l'intégralité de la demande ou de la réponse sans chiffrement, même pour un appel HTTPS. Pour les demandes ou les réponses volumineuses (par exemple, pour télécharger un fichier Amazon S3) ou pour les réponses, l'enregistrement détaillé des connexions peut également avoir un impact significatif sur les performances de votre application.

ApacheHttpClient

Ajoutez l'enregistreur « org.apache.http.wire » au fichier de `log4j2.xml` configuration et réglez le niveau sur « DEBUG ».

Le `log4j2.xml` fichier suivant active la journalisation complète pour Apache HttpClient.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
    </Console>
  </Appenders>

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender"/>
    </Root>
    <Logger name="software.amazon.awssdk" level="WARN" />
  </Loggers>
</Configuration>
```

```

<Logger name="software.amazon.awssdk.request" level="DEBUG" />
<Logger name="org.apache.http.wire" level="DEBUG" />
</Loggers>
</Configuration>

```

Une dépendance Maven supplémentaire à l'égard de `log4j-1.2-api` est requise pour l'enregistrement des connexions avec Apache, car celui-ci utilise la version 1.2 sous le capot.

L'ensemble complet des dépendances Maven pour log4j 2, y compris l'enregistrement des connexions pour le client HTTP Apache, est présenté dans les extraits de fichier de compilation suivants.

Maven

```

...
<dependencyManagement>
  ...
  <dependencies>
    <dependency>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-bom</artifactId>
      <version>VERSION</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
...
<!-- The following is needed for Log4j2 with SLF4J -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-slf4j2-impl</artifactId>
</dependency>

<!-- The following is needed for Apache HttpClient wire logging -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-1.2-api</artifactId>
</dependency>
...

```

Gradle-Kotlin DSL

```

...
dependencies {
    ...
    implementation(platform("org.apache.logging.log4j:log4j-bom:VERSION"))
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")
    implementation("org.apache.logging.log4j:log4j-1.2-api")
}
...

```

2.20.0 À utiliser pour la version minimale de l'log4j-bom artefact. Pour la dernière version, utilisez la version publiée sur [Maven Central](#). Remplacez *VERSION* par la version que vous utiliserez.

URLConnectionHttpClient

Pour enregistrer les informations relatives aux clients du service qui utilisent le `URLConnectionHttpClient`, créez d'abord un `logging.properties` fichier avec le contenu suivant :

```

handlers=java.util.logging.ConsoleHandler
java.util.logging.ConsoleHandler.level=FINEST
sun.net.www.protocol.http.HttpURLConnection.level=ALL

```

Définissez la propriété du système JVM suivante avec le chemin complet de `logging.properties` :

```
-Djava.util.logging.config.file=/full/path/to/logging.properties
```

Cette configuration enregistrera uniquement les en-têtes de la demande et de la réponse, par exemple :

```

<Request> FINE: sun.net.www.MessageHeader@35a9782c11 pairs: {GET /fileuploadtest
HTTP/1.1: null}{amz-sdk-invocation-id: 5f7e707e-4ac5-bef5-ba62-00d71034ffdc}
{amz-sdk-request: attempt=1; max=4}{Authorization: AWS4-HMAC-SHA256
Credential=<deleted>/20220927/us-east-1/s3/aws4_request, SignedHeaders=amz-sdk-
invocation-id;amz-sdk-request;host;x-amz-content-sha256;x-amz-date;x-amz-te,
Signature=e367fa0bc217a6a65675bb743e1280cf12f8e8d566196a816d948fdf0b42ca1a}{User-
Agent: aws-sdk-java/2.17.230 Mac_OS_X/12.5 OpenJDK_64-Bit_Server_VM/25.332-b08
Java/1.8.0_332 vendor/Amazon.com_Inc. io/sync http/URLConnection cfg/retry-mode/
legacy}{x-amz-content-sha256: UNSIGNED-PAYLOAD}{X-Amz-Date: 20220927T133955Z}{x-amz-

```



```
te: append-md5}{Host: tkhill-test1.s3.amazonaws.com}{Accept: text/html, image/gif,
image/jpeg, *, q=.2, */*; q=.2}{Connection: keep-alive}
<Response> FINE: sun.net.www.MessageHeader@70a36a6611 pairs: {null: HTTP/1.1
200 OK}{x-amz-id-2: sAFeZD0KdUMsBbkdjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFu0vr2tUb7Y1wEHGdJ3DSIxq0=}{x-amz-request-id: P9QW9SMZ97FKZ9X7}{Date: Tue,
27 Sep 2022 13:39:57 GMT}{Last-Modified: Tue, 13 Sep 2022 14:38:12 GMT}{ETag:
"2cbe5ad4a064cedec33b452bebf48032"}{x-amz-transfer-encoding: append-md5}{Accept-
Ranges: bytes}{Content-Type: text/plain}{Server: AmazonS3}{Content-Length: 67}
```

Pour voir les corps de demande/réponse, ajoutez-les `-Djavax.net.debug=all` aux propriétés de la JVM. Cette propriété supplémentaire enregistre de nombreuses informations, y compris toutes les informations SSL.

Dans la console ou le fichier journal, recherchez "GET" ou accédez rapidement "POST" à la section du journal contenant les demandes et réponses réelles. "Plaintext before ENCRYPTION" Recherchez des demandes et des réponses "Plaintext after DECRYPTION" pour voir le texte intégral des en-têtes et des corps.

NettyNioAsyncHttpClient

Si votre client de service asynchrone utilise la valeur par défaut `NettyNioAsyncHttpClient`, ajoutez deux enregistreurs supplémentaires à votre `log4j2.xml` fichier pour enregistrer les en-têtes HTTP et les corps de demande/réponse.

```
<Logger name="io.netty.handler.logging" level="DEBUG" />
<Logger name="io.netty.handler.codec.http2.Http2FrameLogger" level="DEBUG" />
```

Voici un `log4j2.xml` exemple complet :

```
<Configuration status="WARN">
  <Appenders>
    <Console name="ConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m
%n" />
    </Console>
  </Appenders>

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender"/>
    </Root>
    <Logger name="software.amazon.awssdk" level="WARN" />
  </Loggers>
</Configuration>
```

```

    <Logger name="software.amazon.awssdk.request" level="DEBUG" />
    <Logger name="io.netty.handler.logging" level="DEBUG" />
    <Logger name="io.netty.handler.codec.http2.Http2FrameLogger" level="DEBUG" /
  >
  </Loggers>
</Configuration>

```

Ces paramètres enregistrent tous les détails de l'en-tête et les corps de demande/réponse.

AwsCrtAsyncHttpClient

Si vous avez configuré votre client de service pour utiliser une instance de `AwsCrtAsyncHttpClient`, vous pouvez enregistrer les détails en définissant les propriétés du système JVM ou par programmation.

Log to a file at "Debug" level

À l'aide des propriétés du système :

```

-Daws.crt.log.level=Trace
-Daws.crt.log.destination=File
-Daws.crt.log.filename=<path to file>

```

Programmatically :

```

import software.amazon.awssdk.crt.Log;

// Execute this statement before constructing the
// SDK service client.
Log.initLoggingToFile(Log.LogLevel.Trace,
    "<path to file>");

```

Log to the console at "Debug" level

À l'aide des propriétés du système :

```

-Daws.crt.log.level=Trace
-Daws.crt.log.destination=Stdout

```

Programmatically :

```

import software.amazon.awssdk.crt.Log;

// Execute this statement before constructing the
// SDK service client.
Log.initLoggingToStdout(Log.LogLevel.Trace);

```

Pour des raisons de sécurité, au niveau « Trace », seuls les `AwsCrtAsyncHttpClient` en-têtes de réponse sont enregistrés. Les en-têtes de demande, les corps de demande et les corps de réponse ne sont pas enregistrés.

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.

Bonnes pratiques pour AWS SDK for Java 2.x

Cette section répertorie les meilleures pratiques relatives à l'utilisation du SDK pour Java 2.x.

Rubriques

- [Réutilisez un client SDK, si possible](#)
- [Fermez les flux d'entrée provenant des opérations du client](#)
- [Ajustez les configurations HTTP en fonction de tests de performance](#)
- [Utiliser OpenSSL pour le client HTTP basé sur Netty](#)
- [Configurer les délais d'expiration de l'API](#)
- [Utiliser les métriques](#)

Réutilisez un client SDK, si possible

Chaque client SDK gère son propre pool de connexions HTTP. Une connexion qui existe déjà dans le pool peut être réutilisée par une nouvelle demande afin de réduire le temps nécessaire à l'établissement d'une nouvelle connexion. Nous vous recommandons de partager une seule instance du client afin d'éviter les surcharges liées à un trop grand nombre de pools de connexions qui ne sont pas utilisés efficacement. Tous les clients du SDK sont compatibles avec les threads.

Si vous ne souhaitez pas partager une instance client, faites appel `close()` à l'instance pour libérer les ressources lorsque le client n'est pas nécessaire.

Fermez les flux d'entrée provenant des opérations du client

Pour les opérations de streaming telles que [S3Client#getObject](#), par exemple, si vous travaillez [ResponseInputStream](#) directement avec, nous vous recommandons de procéder comme suit :

- Lisez toutes les données du flux d'entrée dès que possible.
- Fermez le flux d'entrée dès que possible.

Nous faisons ces recommandations car le flux d'entrée est un flux direct de données provenant de la connexion HTTP et la connexion HTTP sous-jacente ne peut pas être réutilisée tant que toutes les données du flux n'ont pas été lues et que le flux n'est pas fermé. Si ces règles ne sont pas respectées, le client risque de manquer de ressources en allouant un trop grand nombre de connexions HTTP ouvertes mais non utilisées.

Ajustez les configurations HTTP en fonction de tests de performance

Le SDK fournit un ensemble de [configurations HTTP par défaut](#) qui s'appliquent aux cas d'utilisation généraux. Nous recommandons aux clients d'ajuster les configurations HTTP de leurs applications en fonction de leurs cas d'utilisation.

Comme bon point de départ, le SDK propose une fonctionnalité [intelligente de configuration par défaut](#). Cette fonctionnalité est disponible à partir de la version 2.17.102. Vous choisissez un mode en fonction de votre cas d'utilisation, qui fournit des valeurs de configuration pertinentes.

Utiliser OpenSSL pour le client HTTP basé sur Netty

Par défaut, le SDK [NettyNioAsyncHttpClient](#) utilise l'implémentation SSL par défaut du JDK comme `SslProvider`. Nos tests ont révélé qu'OpenSSL fonctionne mieux que l'implémentation par défaut du JDK. La communauté Netty [recommande également d'utiliser OpenSSL](#).

Pour utiliser OpenSSL, `netty-tcnative` complétez vos dépendances. Pour plus de détails sur la configuration, consultez la [documentation du projet Netty](#).

Après avoir `netty-tcnative` configuré votre projet, l'`NettyNioAsyncHttpClient` instance sélectionnera automatiquement OpenSSL. Vous pouvez également définir le `SslProvider` explicitement à l'aide du `NettyNioAsyncHttpClient` générateur, comme indiqué dans l'extrait de code suivant.

```
NettyNioAsyncHttpClient.builder()
    .sslProvider(SslProvider.OPENSSL)
    .build();
```

Configurer les délais d'expiration de l'API

Le SDK fournit des [valeurs par défaut](#) pour certaines options de temporisation, telles que les délais de connexion et les délais d'expiration des sockets, mais pas pour les délais d'expiration des appels

d'API ou les délais d'expiration des tentatives d'appels d'API individuels. Il est recommandé de définir des délais d'expiration à la fois pour les tentatives individuelles et pour l'ensemble de la demande. Cela garantira un échec rapide et optimal de votre application en cas de problèmes transitoires susceptibles de retarder le traitement des demandes ou de provoquer des problèmes réseau fatals.

Vous pouvez configurer des délais d'expiration pour toutes les demandes effectuées par les clients d'un service à l'aide de [ClientOverrideConfiguration#apiCallAttemptTimeout](#) et [ClientOverrideConfiguration#apiCallTimeout](#).

L'exemple suivant montre la configuration d'un client Amazon S3 avec des valeurs de délai d'expiration personnalisées.

```
S3Client.builder()
    .overrideConfiguration(
        b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
            .apiCallAttemptTimeout(Duration.ofMillis(<custom value>)))
    .build();
```

apiCallAttemptTimeout

Ce paramètre définit la durée d'une seule tentative HTTP, après laquelle l'appel d'API peut être réessayé.

apiCallTimeout

La valeur de cette propriété configure la durée de l'exécution complète, y compris toutes les tentatives de nouvelle tentative.

Au lieu de définir ces valeurs de délai d'expiration sur le client de service, vous pouvez utiliser [RequestOverrideConfiguration#apiCallTimeout\(\)](#) et [RequestOverrideConfiguration#apiCallAttemptTimeout\(\)](#) configurer une seule demande.

L'exemple suivant configure une seule `listBuckets` demande avec des valeurs de délai d'expiration personnalisées.

```
s3Client.listBuckets(lbr -> lbr.overrideConfiguration(
    b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
        .apiCallAttemptTimeout(Duration.ofMillis(<custom value>))));
```

Lorsque vous utilisez ces propriétés ensemble, vous fixez une limite stricte au temps total consacré à toutes les tentatives entre les tentatives. Vous pouvez également configurer une requête HTTP individuelle pour qu'elle échoue rapidement en cas de requête lente.

Utiliser les métriques

Le SDK for Java [peut collecter des](#) métriques pour les clients de service de votre application. Vous pouvez utiliser ces indicateurs pour identifier les problèmes de performance, examiner les tendances générales d'utilisation, examiner les exceptions renvoyées par les clients du service ou pour approfondir la compréhension d'un problème particulier.

Nous vous recommandons de collecter des métriques, puis d'analyser les Amazon CloudWatch Logs afin de mieux comprendre les performances de votre application.

Questions fréquentes sur le dépannage

Lorsque vous utilisez le AWS SDK for Java 2.x dans vos applications, vous pouvez rencontrer les erreurs d'exécution répertoriées dans cette rubrique. Utilisez les suggestions présentées ici pour vous aider à découvrir la cause première et à résoudre l'erreur.

Comment corriger l'erreur « **java.net.SocketException** : Réinitialisation de la connexion » ou « Le serveur n'a pas réussi à terminer la réponse » ?

Une erreur de réinitialisation de connexion indique que votre hôte Service AWS, le ou tout autre intermédiaire (par exemple, une passerelle NAT, un proxy, un équilibreur de charge) a fermé la connexion avant que la demande ne soit terminée. Les causes étant nombreuses, pour trouver une solution, vous devez savoir pourquoi la connexion est fermée. Les éléments suivants entraînent généralement la fermeture d'une connexion.

- La connexion est inactive. Cela est courant pour les opérations de streaming, lorsque les données ne sont pas écrites sur ou depuis le fil pendant un certain temps, de sorte qu'un intermédiaire détecte que la connexion est morte et la ferme. Pour éviter cela, assurez-vous que votre application télécharge ou télécharge activement des données.
- Vous avez fermé le client HTTP ou SDK. Veillez à ne pas fermer les ressources lorsqu'elles sont en cours d'utilisation.

- Un proxy mal configuré. Essayez de contourner les proxys que vous avez configurés pour voir si cela résout le problème. Si cela résout le problème, le proxy ferme votre connexion pour une raison ou une autre. Recherchez votre proxy spécifique pour déterminer pourquoi il ferme la connexion.

Si vous ne parvenez pas à identifier le problème, essayez d'exécuter un vidage TCP pour une connexion affectée à la périphérie client de votre réseau (par exemple, après tous les proxys que vous contrôlez).

Si vous constatez que le AWS terminal envoie une TCP RST (réinitialisation), [contactez le service concerné](#) pour savoir s'il peut déterminer la raison de la réinitialisation. Soyez prêt à fournir des identifiants de demande et des horodatages indiquant le moment où le problème s'est produit. L'équipe d' AWS assistance peut également bénéficier de [journaux de connexion](#) indiquant exactement quels octets votre application envoie et reçoit et à quel moment.

Comment corriger le « délai d'attente de connexion » ?

Une erreur de délai de connexion indique que votre hôte Service AWS, le ou tout autre intermédiaire (par exemple, une passerelle NAT, un proxy, un équilibreur de charge) n'a pas réussi à établir une nouvelle connexion avec le serveur dans le délai de connexion configuré. Les éléments suivants décrivent les causes courantes de ce problème.

- Le délai d'expiration de connexion configuré est trop faible. Par défaut, le délai de connexion est de 2 secondes dans le AWS SDK for Java 2.x. Si vous définissez un délai de connexion trop faible, cette erreur peut s'afficher. Le délai de connexion recommandé est de 1 seconde si vous effectuez uniquement des appels régionaux et de 3 secondes si vous effectuez des demandes interrégionales.
- Un proxy mal configuré. Essayez de contourner les proxys que vous avez configurés pour voir si cela résout le problème. Si cela résout le problème, le proxy est à l'origine du délai d'expiration de la connexion. Recherchez votre proxy spécifique pour déterminer pourquoi cela se produit

Si vous ne parvenez pas à identifier le problème, essayez d'exécuter un vidage TCP pour une connexion affectée à la périphérie client de votre réseau (par exemple, après tout proxy que vous contrôlez) afin d'étudier tout problème réseau.

Comment corriger « `java.net.SocketTimeoutException` : Expiration du délai de lecture » ?

Une erreur de délai de lecture indique que la machine virtuelle Java a tenté de lire les données du système d'exploitation sous-jacent, mais que les données n'ont pas été renvoyées dans le délai configuré via le SDK. Cette erreur peut se produire si le système d'exploitation Service AWS, ou tout autre intermédiaire (par exemple, une passerelle NAT, un proxy, un équilibreur de charge) n'envoie pas de données dans le délai prévu par la JVM. Les causes étant nombreuses, pour trouver une solution, vous devez savoir pourquoi les données ne sont pas renvoyées.

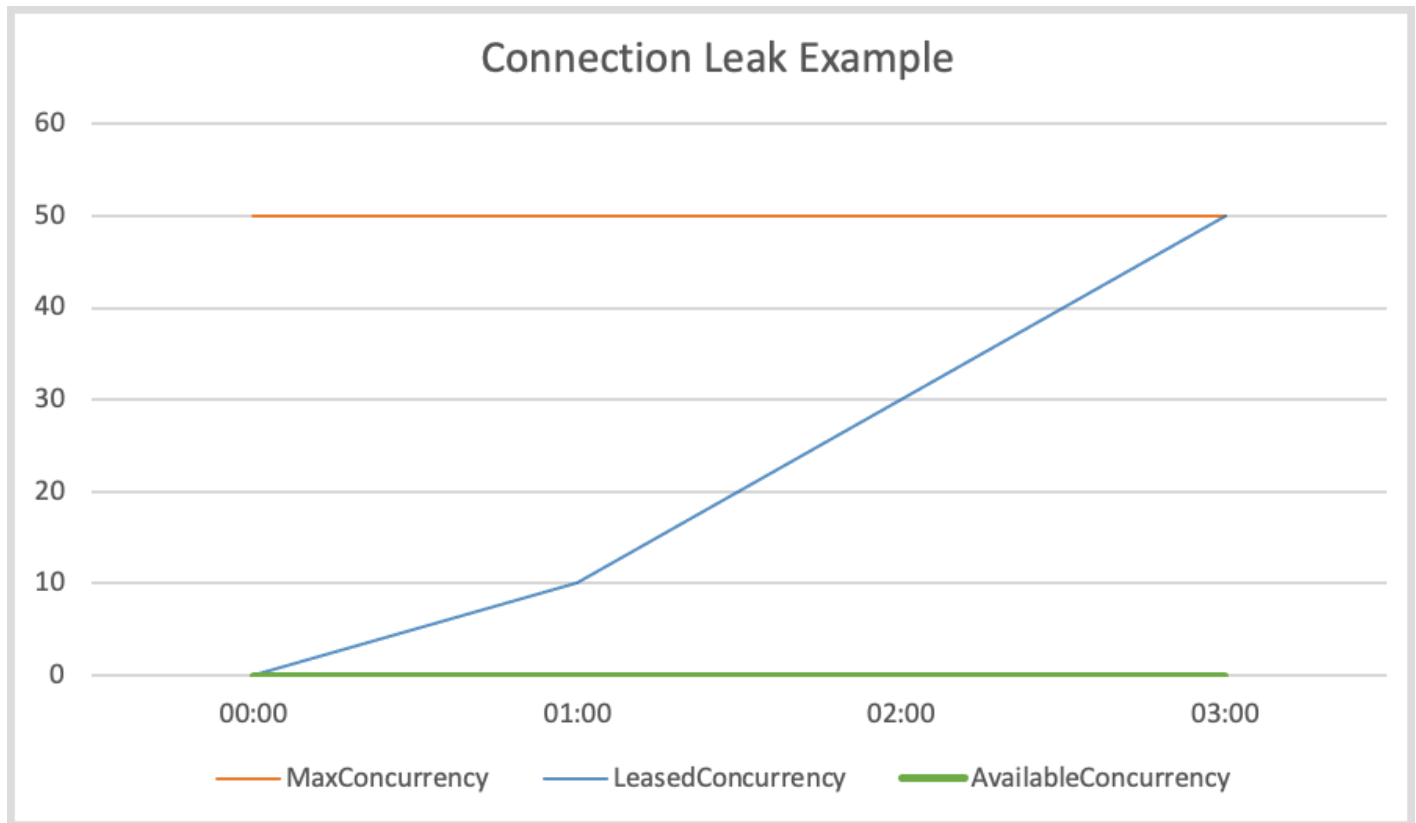
Essayez d'exécuter un vidage TCP pour une connexion affectée à la périphérie client de votre réseau (par exemple, après tous les proxys que vous contrôlez).

Si vous constatez que le AWS terminal envoie une TCP RST (réinitialisation), [contactez le service concerné](#). Soyez prêt à fournir des identifiants de demande et des horodatages indiquant le moment où le problème s'est produit. L'équipe d' AWS assistance peut également bénéficier de [journaux de connexion](#) indiquant exactement quels octets votre application envoie et reçoit et à quel moment.

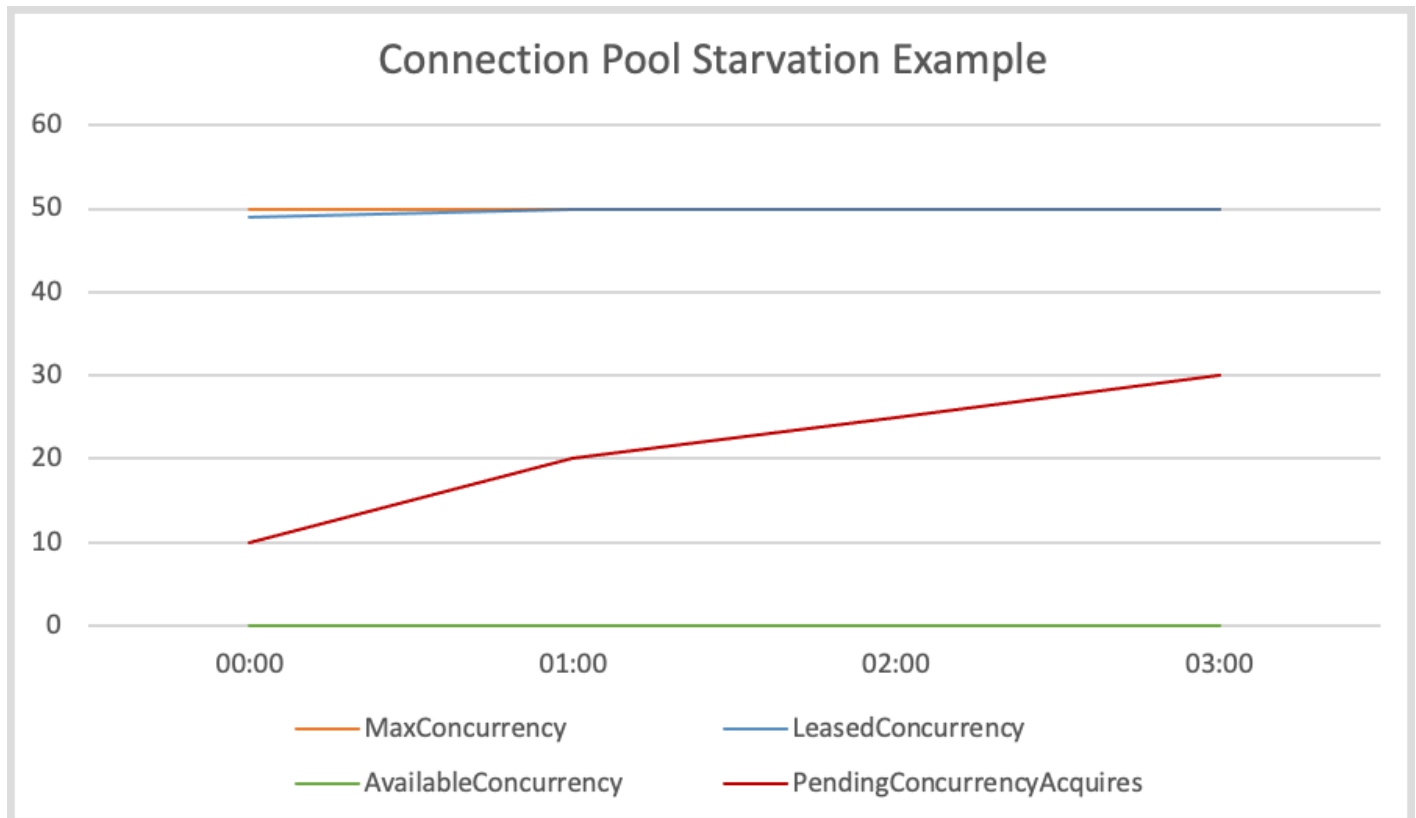
Comment corriger l'erreur « Impossible d'exécuter la requête HTTP : délai d'attente de connexion depuis le pool » ?

Cette erreur indique qu'une demande ne peut pas obtenir de connexion depuis le pool dans le délai maximum spécifié. Pour résoudre ce problème, nous vous recommandons d'[activer les métriques côté client du SDK pour publier des métriques](#) sur Amazon CloudWatch. Les métriques HTTP peuvent aider à identifier la cause première. Les éléments suivants décrivent les causes courantes de cette erreur.

- Fuite de connexion. Vous pouvez étudier cela en cochant `LeasedConcurrencyAvailableConcurrency`, et en utilisant `MaxConcurrency` des métriques. S'il `LeasedConcurrency` augmente jusqu'à atteindre `MaxConcurrency` mais ne diminue jamais, il se peut qu'il y ait une fuite de connexion. Une fuite est souvent due au fait qu'une opération de streaming, telle qu'une `getObject` méthode S3, n'est pas fermée. Nous recommandons à votre application de lire toutes les données du flux d'entrée dès que possible et de [fermer le flux d'entrée par la suite](#). Le graphique suivant montre à quoi peuvent ressembler les métriques du SDK en cas de fuite de connexion.



- Inanition du pool de connexion. Cela peut se produire si votre taux de demandes est trop élevé et que la taille du pool de connexions configuré ne peut pas répondre à la demande. La taille du pool de connexions par défaut est de 50, et lorsque les connexions du pool atteignent leur maximum, le client HTTP met les demandes entrantes en file d'attente jusqu'à ce que les connexions soient disponibles. Le graphique suivant montre à quoi peuvent ressembler les métriques du SDK en cas d'indisponibilité du pool de connexions.



Pour atténuer ce problème, envisagez de prendre l'une des mesures suivantes.

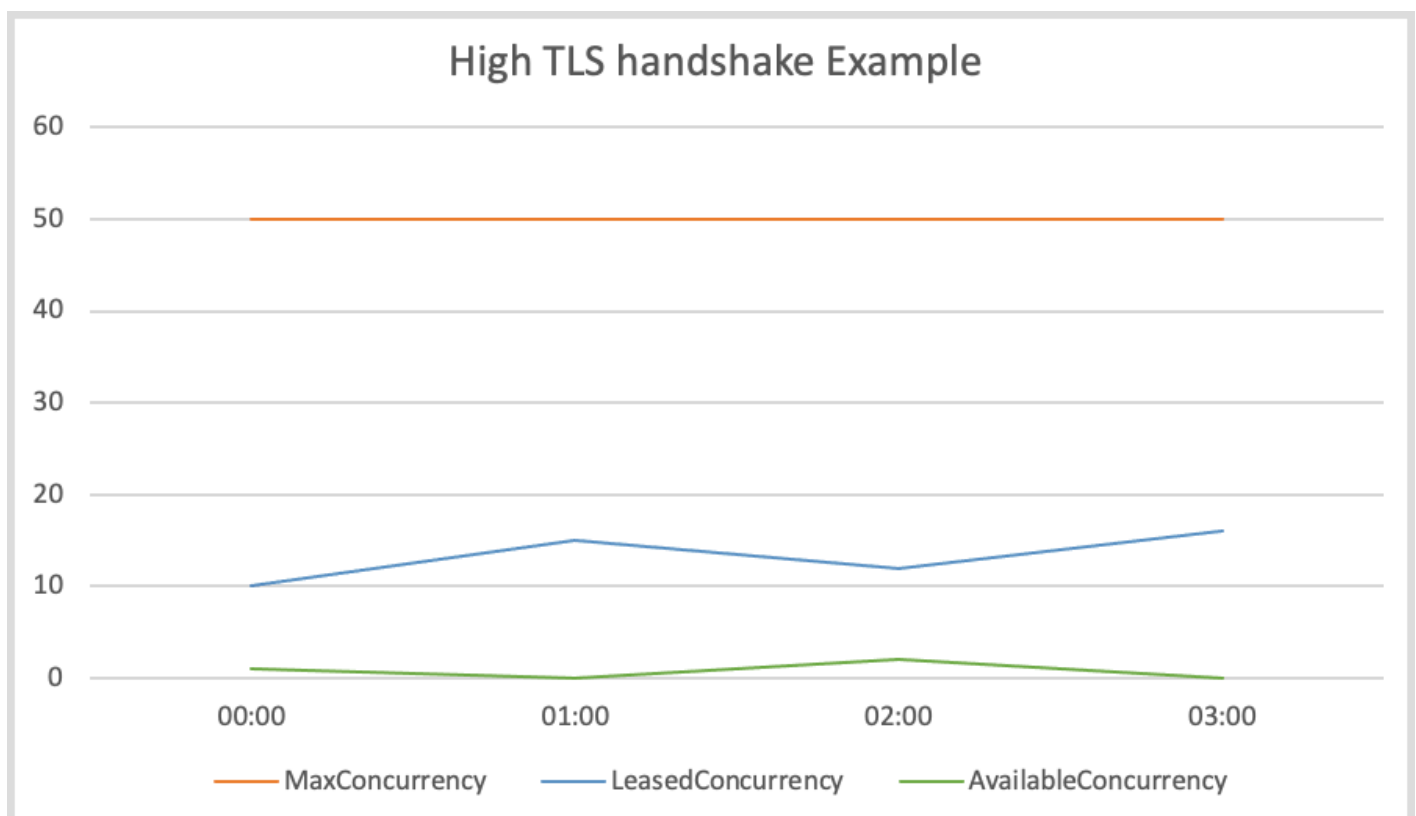
- Augmentez la taille du pool de connexions,
- Augmentez le délai d'acquisition.
- Ralentissez le taux de demandes.

En augmentant le nombre maximum de connexions, le débit du client peut augmenter (sauf si l'interface réseau est déjà pleinement utilisée). Cependant, vous pouvez éventuellement atteindre les limites du système d'exploitation concernant le nombre de descripteurs de fichiers utilisés par le processus. Si vous utilisez déjà pleinement votre interface réseau ou si vous ne parvenez pas à augmenter davantage le nombre de connexions, essayez d'augmenter le délai d'acquisition. Avec cette augmentation, vous gagnez du temps supplémentaire pour les demandes visant à établir une connexion avant l'expiration du délai imparti. Si les connexions ne se libèrent pas, les demandes suivantes expireront toujours.

Si vous ne parvenez pas à résoudre le problème en utilisant les deux premiers mécanismes, ralentissez le taux de demandes en essayant les options suivantes.

- Simplifiez vos demandes afin que les fortes rafales de trafic ne surchargent pas le client.
- Soyez plus efficace avec les appels vers Services AWS.

- Augmentez le nombre d'hôtes qui envoient des demandes.
- Les threads d'E/S sont trop occupés. Cela ne s'applique que si vous utilisez un client SDK asynchrone avec [NettyNioAsyncHttpClient](#). Si la `AvailableConcurrency` métrique n'est pas faible, ce qui indique que des connexions sont disponibles dans le pool, mais que `ConcurrencyAcquireDuration` est élevée, cela peut être dû au fait que les threads d'E/S ne sont pas en mesure de traiter les demandes. Assurez-vous de ne pas vous faire passer `Runnable::run` pour un [futur exécuteur de complétion](#) et d'exécuter une tâche fastidieuse dans la chaîne de complétion future de la réponse, car cela peut bloquer un thread d'E/S. Si ce n'est pas le cas, envisagez d'augmenter le nombre de threads d'E/S en utilisant [eventLoopGroupBuilder](#) cette méthode. À titre de référence, le nombre par défaut de threads d'E/S pour une `NettyNioAsyncHttpClient` instance est le double du nombre de cœurs de processeur de l'hôte.
- Latence de connexion TLS élevée. Si votre `AvailableConcurrency` métrique est proche de 0 mais inférieure à `MaxConcurrency`, cela peut être dû au fait que la latence du handshake TLS est élevée. Le graphique suivant montre à quoi peuvent ressembler les métriques du SDK en cas de latence élevée en matière de prise de contact TLS.



Pour les clients HTTP proposés par le SDK Java qui ne sont pas basés sur le CRT, essayez d'activer les [journaux TLS pour résoudre les problèmes liés au TLS](#). Pour le client HTTP AWS

basé sur CRT, essayez d'activer les journaux [AWS CRT](#). Si vous constatez que le AWS point de terminaison semble mettre du temps à effectuer une prise de contact TLS, vous devez [contacter le service concerné](#).

Comment réparer un `NoClassDefFoundError`, `NoSuchMethodError` ou `NoSuchFieldError` ?

A `NoClassDefFoundError` indique qu'une classe n'a pas pu être chargée lors de l'exécution. Les deux causes les plus fréquentes de cette erreur sont les suivantes :

- la classe n'existe pas dans le chemin de classe car le fichier JAR est absent ou la mauvaise version du fichier JAR se trouve sur le chemin de classe.
- la classe n'a pas pu être chargée car son initialiseur statique a généré une exception.

De même, `NoSuchMethodError` s et `NoSuchFieldError` s résultent généralement d'une version de JAR incompatible. Nous vous recommandons de suivre les étapes suivantes.

1. Vérifiez vos dépendances pour vous assurer que vous utilisez la même version de tous les fichiers JAR du SDK. La raison la plus courante pour laquelle une classe, une méthode ou un champ est introuvable est lorsque vous effectuez une mise à niveau vers une nouvelle version du client mais que vous continuez à utiliser une ancienne version de dépendance « partagée » du SDK. La nouvelle version du client peut tenter d'utiliser des classes qui n'existent que dans les nouvelles dépendances « partagées » du SDK. Essayez d'exécuter `mvn dependency:tree` ou `gradle dependencies` (pour Gradle) de vérifier que les versions de la bibliothèque du SDK correspondent toutes. Pour éviter complètement ce problème à l'avenir, nous vous recommandons d'utiliser [BOM \(Bill of Materials\)](#) pour gérer les versions des modules du SDK.

L'exemple suivant montre un exemple de versions mixtes du SDK.

```
[INFO] +- software.amazon.awssdk:dynamodb:jar:2.20.00:compile
[INFO] |   +- software.amazon.awssdk:aws-core:jar:2.13.19:compile
[INFO] +- software.amazon.awssdk:netty-nio-client:jar:2.20.00:compile
```

La version de `dynamodb` est 2.20.00 et la version de `aws-core` est 2.13.19. La version de l'`aws-core` artefact doit également être 2.20.00.

2. Vérifiez les instructions au début de vos journaux pour voir si une classe ne parvient pas à se charger en raison d'un échec d'initialisation statique. La première fois que la classe ne se charge pas, elle peut générer une exception différente, plus utile, qui indique pourquoi la classe ne peut pas être chargée. Cette exception potentiellement utile ne se produit qu'une seule fois, de sorte que les instructions de journal ultérieures indiqueront uniquement que la classe est introuvable.
3. Vérifiez votre processus de déploiement pour vous assurer qu'il déploie réellement les fichiers JAR requis en même temps que votre application. Il est possible que vous construisiez avec la bonne version, mais le processus qui crée le chemin de classe pour votre application exclut une dépendance requise.

Comment corriger une erreur « » ou une erreur **SignatureDoesNotMatch** « La signature de demande que nous avons calculée ne correspond pas à la signature que vous avez fournie » ?

Une `SignatureDoesNotMatch` erreur indique que la signature générée par le AWS SDK for Java et celle générée par le Service AWS ne correspondent pas. Les éléments suivants décrivent les causes potentielles.

- Un mandataire ou un intermédiaire modifie la demande. Par exemple, un proxy ou un équilibreur de charge peut modifier un en-tête, un chemin ou une chaîne de requête signés par le SDK.
- Le service et le SDK diffèrent dans la manière dont ils encodent la demande lorsque chacun génère la chaîne à signer.

Pour résoudre ce problème, nous vous recommandons d'[activer la journalisation du débogage](#) pour le SDK. Essayez de reproduire l'erreur et de trouver la demande canonique générée par le SDK. Dans le journal, la demande canonique est étiquetée avec `AWS4 Canonical Request: ...` et la chaîne à signer est étiquetée `AWS4 String to sign:`

Si vous ne pouvez pas activer le débogage, par exemple parce qu'il n'est reproductible qu'en production, ajoutez à votre application une logique qui enregistre les informations relatives à la demande lorsque l'erreur se produit. Vous pouvez ensuite utiliser ces informations pour essayer de reproduire l'erreur en dehors de la production lors d'un test d'intégration avec la journalisation du débogage activée.

Après avoir collecté la demande canonique et la chaîne à signer, comparez-les à la [spécification AWS Signature Version 4](#) pour déterminer s'il existe des problèmes dans la manière dont le SDK

a généré la chaîne à signer. Si quelque chose ne va pas, vous pouvez créer un [rapport de GitHub](#) [bogue](#) auprès du AWS SDK for Java.

Si aucun problème ne s'affiche, vous pouvez comparer la chaîne à signer du SDK avec la chaîne à signer que certains Services AWS renvoient dans le cadre de la réponse à l'échec (Amazon S3, par exemple). Si ce n'est pas disponible, vous devez [contacter le service concerné](#) pour voir quelle demande canonique et quelle chaîne de signature ils ont générées à des fins de comparaison. Ces comparaisons peuvent aider à identifier les parties intermédiaires susceptibles d'avoir modifié la demande ou les différences de codage entre le service et le client.

Pour plus d'informations générales sur les demandes de signature, consultez la section [Signature des demandes d' AWS API](#) dans le guide de AWS Identity and Access Management l'utilisateur.

Exemple d'une demande canonique

```
PUT
/Example-Bucket/Example-Object
partNumber=19&uploadId=string
amz-sdk-invocation-id:f8c2799d-367c-f024-e8fa-6ad6d0a1afb9
amz-sdk-request:attempt=1; max=4
content-encoding:aws-chunked
content-length:51
content-type:application/octet-stream
host:xxxxx
x-amz-content-sha256:STREAMING-UNSIGNED-PAYLOAD-TRAILER
x-amz-date:20240308T034733Z
x-amz-decoded-content-length:10
x-amz-sdk-checksum-algorithm:CRC32
x-amz-trailer:x-amz-checksum-crc32
```

Exemple d'une chaîne à signer

```
AWS4-HMAC-SHA256
20240308T034435Z
20240308/us-east-1/s3/aws4_request
5f20a7604b1ef65dd89c333fd66736fdef9578d11a4f5d22d289597c387dc713
```

Comment corriger l'erreur « `java.lang.IllegalStateException` : arrêt du pool de connexions » ?

Cette erreur indique que le pool de connexions HTTP Apache sous-jacent a été fermé. Les éléments suivants décrivent les causes potentielles.

- Le client SDK a été fermé prématurément. Le SDK ferme le pool de connexions uniquement lorsque le client associé est fermé. Veillez à ne pas fermer les ressources lorsqu'elles sont en cours d'utilisation.
- Un `java.lang.Error` a été lancé. Des erreurs telles que `OutOfMemoryError` l'[arrêt](#) d'un pool de connexions HTTP Apache. Examinez vos journaux à la recherche de traces d'erreurs. Vérifiez également dans votre code les endroits où il détecte `Throwable` s ou `Error` s mais en absorbe le résultat, ce qui empêche l'erreur de réapparaître. Si votre code ne signale aucune erreur, réécrivez-le afin que les informations soient enregistrées. Les informations enregistrées permettent de déterminer la cause première de l'erreur.
- Vous avez tenté d'utiliser le fournisseur d'informations d'identification renvoyé `DefaultCredentialsProvider#create()` après sa fermeture. [DefaultCredentialsProvider#create](#) renvoie une instance de singleton. Ainsi, si elle est fermée et que votre code appelle la `resolveCredentials` méthode, l'exception est déclenchée après l'expiration des informations d'identification (ou jeton) mises en cache.

Vérifiez dans votre code les endroits où le `DefaultCredentialsProvider` est fermé, comme indiqué dans les exemples suivants.

- L'instance singleton est fermée en appelant `DefaultCredentialsProvider#close()`.

```
DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create(); // Singleton instance returned.
AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();

// Make calls to Services AWS.

defaultCredentialsProvider.close(); // Explicit close.

// Make calls to Services AWS.

// After the credentials expire, either of the following calls eventually results
// in a "Connection pool shut down" exception.
credentials = defaultCredentialsProvider.resolveCredentials();
// Or
```



```
credentials = DefaultCredentialsProvider.create().resolveCredentials();
```

- Invoquez `DefaultCredentialsProvider#create()` dans un try-with-resources bloc.

```
try (DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create()) {
    AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();

    // Make calls to Services AWS.

} // After the try-with-resources block exits, the singleton
DefaultCredentialsProvider is closed.

// Make calls to Services AWS.

DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create(); // The closed singleton instance is returned.
// If the credentials (or token) has expired, the following call results in the
error.
AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();
```

Créez une nouvelle instance non singleton en appelant `DefaultCredentialsProvider.builder().build()` si votre code a fermé l'instance singleton et que vous devez résoudre les informations d'identification à l'aide d'un `DefaultCredentialsProvider`

Utilisez les fonctionnalités de la version AWS SDK for Java 2.x

Caractéristiques générales

Le SDK pour Java 2.x contient plusieurs fonctionnalités qui facilitent la programmation Services AWS .

- Le SDK masque les mécanismes complexes qui sous-tendent la [récupération des résultats paginés](#) et l'[interrogation](#) des ressources.
- La [programmation asynchrone avec E/S non bloquantes vous permet d'écrire](#) du code simultané avec de meilleures performances. Le SDK offre les avantages du [HTTP/2](#), tels qu'une latence réduite, dans la mesure du possible.
- Le SDK Java peut générer des [métriques](#) pour vous aider à surveiller l'état de fonctionnement de vos applications.

Fonctionnalités spécifiques au service

Outre les fonctionnalités générales mentionnées précédemment, le SDK Java fournit des fonctionnalités spécifiques Services AWS.

- Amazon S3 — Pour [simplifier votre travail avec les fichiers et les répertoires](#) avec Amazon S3, le SDK fournit le gestionnaire de transfert S3. Pour [améliorer les performances et la fiabilité](#) lors de l'utilisation de l'API S3 asynchrone standard du SDK, le SDK propose le AWS client S3 basé sur CRT.
- DynamoDB — [La fonctionnalité de mappage orientée objet est fournie par l'API](#) DynamoDB Enhanced Client. [Travaillez avec des données orientées document de style JSON à l'aide de l'API](#) Enhanced Document.
- IAM — L'API IAM Policy Builder fournit un [moyen sécurisé et orienté objet pour créer](#) des politiques IAM.

Travaillez avec des résultats paginés à l'aide de la version 2.x AWS SDK for Java

De nombreuses AWS opérations renvoient des résultats paginés lorsque l'objet de réponse est trop volumineux pour être renvoyé en une seule réponse. Dans la AWS SDK for Java version 1.0, la réponse contient un jeton que vous utilisez pour récupérer la page de résultats suivante. En revanche, la version AWS SDK for Java 2.x dispose de méthodes de pagination automatique qui effectuent plusieurs appels de service pour obtenir automatiquement la page de résultats suivante pour vous. Il vous suffit d'écrire le code qui traite les résultats. La pagination automatique est disponible pour les clients synchrones et asynchrones.

Note

Ces extraits de code supposent que vous comprenez [les bases de l'utilisation du SDK](#) et que vous avez configuré votre environnement avec [un accès par authentification unique](#).

Pagination synchrone

Les exemples suivants illustrent les méthodes de pagination synchrone permettant de répertorier les objets d'un Amazon S3 compartiment.

Itérer sur les pages

Le premier exemple montre l'utilisation d'un objet `ListRes` paginateur, une [ListObjectsV2Iterable](#) instance, pour parcourir toutes les pages de réponse avec la méthode `stream`. Le code circule sur les pages de réponse, convertit le flux de réponse en flux de [S3Object](#) contenu, puis traite le contenu de l'Amazon S3 objet.

Les importations suivantes s'appliquent à tous les exemples de cette section de pagination synchrone.

Importations

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

```
ListObjectsV2Request listReq = ListObjectsV2Request.builder()
    .bucket(bucketName)
    .maxKeys(1)
    .build();

ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
// Process response pages
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out
        .println(" Key: " + content.key() + "
size = " + content.size()));
```

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

Itérer sur des objets

Les exemples suivants montrent comment itérer sur les objets retournés dans la réponse à la place des pages de la réponse. La méthode `contents` de la classe `ListObjectsV2Iterable` renvoie un [SdkIterable](#) qui fournit plusieurs méthodes pour traiter les éléments de contenu sous-jacents.

Utiliser un stream

L'extrait suivant utilise la `stream` méthode sur le contenu de la réponse pour itérer sur la collection d'éléments paginés.

```
// Helper method to work with paginated collection of items directly.
listRes.contents().stream()
    .forEach(content -> System.out
        .println(" Key: " + content.key() + "
size = " + content.size()));
```

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

Utilisez une boucle pour chaque

Comme l'`SdkIterableIterable` interface est étendue, vous pouvez traiter le contenu comme n'importe quel autre `Iterable`. L'extrait de code suivant utilise `for-each` une boucle standard pour parcourir le contenu de la réponse.

```
for (S3Object content : listRes.contents()) {
    System.out.println(" Key: " + content.key() + " size = " +
content.size());
}
```

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

Pagination manuelle

Si votre cas d'utilisation le requiert, la pagination manuelle demeure disponible. Utilisez le jeton suivant de l'objet de réponse pour les demandes suivantes. L'exemple suivant utilise une `while` boucle.

```
ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
```

```
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    boolean done = false;
    while (!done) {
        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
        for (S3Object content : listObjResponse.contents()) {
            System.out.println(content.key());
        }

        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()

.continuationToken(listObjResponse.nextContinuationToken())
                    .build();
    }
}
```

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

Pagination asynchrone

Les exemples suivants illustrent les méthodes de pagination asynchrone pour répertorier les tables. DynamoDB

Itérer sur les pages des noms de tables

Les deux exemples suivants utilisent un client DynamoDB asynchrone qui appelle la `listTablesPaginator` méthode avec une demande pour obtenir un [ListTablesPublisher](#). `ListTablesPublisher` implémente deux interfaces, qui offrent de nombreuses options pour traiter les réponses. Nous allons examiner les méthodes de chaque interface.

Utilisez un **Subscriber**

L'exemple de code suivant montre comment traiter les résultats paginés à l'aide de `org.reactivestreams.Publisher` interface implémentée par `ListTablesPublisher`. Pour en savoir plus sur le modèle de flux réactifs, consultez le [GitHub référentiel Reactive Streams](#).

Les importations suivantes s'appliquent à tous les exemples de cette section de pagination asynchrone.

Importations

```
import io.reactivex.rxjava3.core.Flowable;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import reactor.core.publisher.Flux;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginators.ListTablesPublisher;

import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
```

Le code suivant permet d'acquérir une `ListTablesPublisher` instance.

```
// Creates a default client with credentials and region loaded from the
// environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(listTablesRequest);
```

Le code suivant utilise une implémentation anonyme de `org.reactivestreams.Subscriber` pour traiter les résultats de chaque page.

La méthode `onSubscribe` appelle la méthode `Subscription.request` pour initier les demandes de données de l'éditeur. Cette méthode doit être appelée pour commencer à obtenir les données à partir de l'éditeur.

La `onNext` méthode de l'abonné traite une page de réponse en accédant à tous les noms de tables et en les imprimant. Une fois la page traitée, une autre page est demandée à l'éditeur. Cette méthode est appelée à plusieurs reprises jusqu'à ce que toutes les pages soient récupérées.

La méthode `onError` est déclenchée si une erreur se produit lors de la récupération des données. Enfin, la méthode `onComplete` est appelée lorsque toutes les pages ont été demandées.

```
    // A Subscription represents a one-to-one life-cycle of a Subscriber
    subscribing
    // to a Publisher.
    publisher.subscribe(new Subscriber<ListTablesResponse>() {
        // Maintain a reference to the subscription object, which is required to
        request
        // data from the publisher.
        private Subscription subscription;

        @Override
        public void onSubscribe(Subscription s) {
            subscription = s;
            // Request method should be called to demand data. Here we request a
            single
            // page.
            subscription.request(1);
        }

        @Override
        public void onNext(ListTablesResponse response) {
            response.tableNames().forEach(System.out::println);
            // After you process the current page, call the request method to
            signal that
            // you are ready for next page.
            subscription.request(1);
        }

        @Override
        public void onError(Throwable t) {
            // Called when an error has occurred while processing the requests.
        }

        @Override
        public void onComplete() {
            // This indicates all the results are delivered and there are no more
            pages
            // left.
        }
    });
```


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

Utilisez un **Consumer**

L'`SdkPublisher` interface qui `ListTablesPublisher` implémente possède une `subscribe` méthode qui prend un `Consumer` et renvoie un `CompletableFuture<Void>`.

La `subscribe` méthode de cette interface peut être utilisée pour des cas d'utilisation simples où une surcharge `org.reactivestreams.Subscriber` peut être trop importante. Comme le code ci-dessous occupe chaque page, il appelle la `tableNames` méthode sur chacune d'elles. La `tableNames` méthode renvoie `java.util.List` l'un des noms de tables DynamoDB traités avec la méthode `forEach`

```
// Use a Consumer for simple use cases.
CompletableFuture<Void> future = publisher.subscribe(
    response -> response.tableNames()
        .forEach(System.out::println));
```

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

Itérer sur les noms des tables

Les exemples suivants montrent comment itérer sur les objets retournés dans la réponse à la place des pages de la réponse. À l'instar de l'exemple synchrone d'Amazon S3 présenté précédemment avec sa `contents` méthode, la classe `ListTablesPublisher` de résultats asynchrone DynamoDB possède `tableNames` la méthode pratique pour interagir avec la collection d'éléments sous-jacente. Le type de retour de la `tableNames` méthode est un [SdkPublisher](#) qui peut être utilisé pour demander des éléments sur toutes les pages.

Utilisez un **Subscriber**

Le code suivant acquiert `SdkPublisher` l'un des ensembles sous-jacents de noms de tables.

```
// Create a default client with credentials and region loaded from the
// environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher listTablesPublisher =
asyncClient.listTablesPaginator(listTablesRequest);
```

```
SdkPublisher<String> publisher = listTablesPublisher.tableNames();
```

Le code suivant utilise une implémentation anonyme de `org.reactivestreams.Subscriber` pour traiter les résultats de chaque page.

La `onNext` méthode de l'abonné traite un élément individuel de la collection. Dans ce cas, il s'agit d'un nom de table. Une fois le nom de table traité, un autre nom de table est demandé à l'éditeur. Cette méthode est appelée à plusieurs reprises jusqu'à ce que tous les noms de table soient récupérés.

```
// Use a Subscriber.
publisher.subscribe(new Subscriber<String>() {
    private Subscription subscription;

    @Override
    public void onSubscribe(Subscription s) {
        subscription = s;
        subscription.request(1);
    }

    @Override
    public void onNext(String tableName) {
        System.out.println(tableName);
        subscription.request(1);
    }

    @Override
    public void onError(Throwable t) {
    }

    @Override
    public void onComplete() {
    }
});
```

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

Utilisez un **Consumer**

L'exemple suivant utilise la `subscribe` méthode `SdkPublisher` qui prend un `Consumer` pour traiter chaque élément.

```
// Use a Consumer.  
CompletableFuture<Void> future = publisher.subscribe(System.out::println);  
future.get();
```

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

Utiliser une bibliothèque tierce

Vous pouvez utiliser d'autres bibliothèques tierces au lieu d'implémenter un abonné personnalisé. Cet exemple illustre l'utilisation de RxJava, mais n'importe quelle bibliothèque implémentant les interfaces de flux réactives peut être utilisée. Consultez la [page RxJava wiki GitHub](#) pour plus d'informations sur cette bibliothèque.

Pour utiliser la bibliothèque, ajoutez-la en tant que dépendance. Si vous utilisez Maven, l'exemple illustre l'extrait POM à utiliser.

Entrée POM

```
<dependency>  
  <groupId>io.reactivex.rxjava3</groupId>  
  <artifactId>rxjava</artifactId>  
  <version>3.1.6</version>  
</dependency>
```

Code

```
DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();  
ListTablesPublisher publisher =  
asyncClient.listTablesPaginator(ListTablesRequest.builder()  
    .build());  
  
// The Flowable class has many helper methods that work with  
// an implementation of an org.reactivestreams.Publisher.  
List<String> tables = Flowable.fromPublisher(publisher)  
    .flatMapIterable(ListTablesResponse::tableNames)  
    .toList()  
    .blockingGet();  
System.out.println(tables);
```

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

Sondage sur l'état des ressources dans la version AWS SDK for Java 2.x : Waiters

L'utilitaire Waiters de la version AWS SDK for Java 2.x vous permet de vérifier que les AWS ressources sont dans un état spécifié avant d'effectuer des opérations sur ces ressources.

Un serveur est une abstraction utilisée pour interroger AWS des ressources, telles que des DynamoDB tables ou des Amazon S3 compartiments, jusqu'à ce qu'un état souhaité soit atteint (ou jusqu'à ce qu'il soit déterminé que la ressource n'atteindra jamais l'état souhaité). Au lieu d'écrire une logique pour interroger continuellement vos AWS ressources, ce qui peut être fastidieux et source d'erreurs, vous pouvez utiliser des serveurs pour interroger une ressource et faire en sorte que votre code continue de s'exécuter une fois que la ressource est prête.

Prérequis

Avant de pouvoir utiliser des serveurs dans un projet avec le AWS SDK for Java, vous devez suivre les étapes décrites dans [Configuration de la version AWS SDK for Java 2.x](#).

Vous devez également configurer les dépendances de votre projet (par exemple, dans votre `build.gradle` fichier `pom.xml` ou dans votre fichier) pour utiliser la version `2.15.0` ou une version ultérieure du AWS SDK for Java.

Par exemple :

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.15.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

Utiliser des serveurs

Pour instancier un objet de serveur, créez d'abord un client de service. Définissez la `waiter()` méthode du client de service comme valeur de l'objet serveur. Une fois que l'instance de serveur existe, définissez ses options de réponse pour exécuter le code approprié.

Programmation synchrone

L'extrait de code suivant montre comment attendre qu'une DynamoDB table existe et passe à l'état ACTIF.

```
DynamoDbClient dynamo = DynamoDbClient.create();
DynamoDbWaiter waiter = dynamo.waiter();

WaiterResponse<DescribeTableResponse> waiterResponse =
    waiter.waitUntilTableExists(r -> r.tableName("myTable"));

// print out the matched response with a tableStatus of ACTIVE
waiterResponse.matched().response().ifPresent(System.out::println);
```

Programmation asynchrone

L'extrait de code suivant montre comment attendre qu'une DynamoDB table n'existe plus.

```
DynamoDbAsyncClient asyncDynamo = DynamoDbAsyncClient.create();
DynamoDbAsyncWaiter asyncWaiter = asyncDynamo.waiter();

CompletableFuture<WaiterResponse<DescribeTableResponse>> waiterResponse =
    asyncWaiter.waitUntilTableNotExists(r -> r.tableName("myTable"));

waiterResponse.whenComplete((r, t) -> {
    if (t == null) {
        // print out the matched ResourceNotFoundException
        r.matched().exception().ifPresent(System.out::println);
    }
}).join();
```

Configuration des serveurs

Vous pouvez personnaliser la configuration d'un serveur en utilisant `overrideConfiguration()` le générateur. Pour certaines opérations, vous pouvez appliquer une configuration personnalisée lorsque vous faites la demande.

Configuration d'un serveur

L'extrait de code suivant montre comment modifier la configuration d'un serveur.

```
// sync
DynamoDbWaiter waiter =
    DynamoDbWaiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(10))
        .client(dynamoDbClient)
        .build();

// async
DynamoDbAsyncWaiter asyncWaiter =
    DynamoDbAsyncWaiter.builder()
        .client(dynamoDbAsyncClient)
        .overrideConfiguration(o -> o.backoffStrategy(
            FixedDelayBackoffStrategy.create(Duration.ofSeconds(2))))
        .scheduledExecutorService(Executors.newScheduledThreadPool(3))
        .build();
```

Configuration de remplacement pour une demande spécifique

L'extrait de code suivant montre comment remplacer la configuration d'un serveur par demande. Notez que seules certaines opérations ont des configurations personnalisables.

```
waiter.waitUntilTableNotExists(b -> b.tableName("myTable"),
    o -> o.maxAttempts(10));

asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),
    o -> o.waitTimeout(Duration.ofMinutes(1)));
```

Exemples de code

Pour un exemple complet d'utilisation de serveurs avec DynamoDB, consultez [CreateTable.java](#) dans le référentiel d'exemples de AWS code.

Pour un exemple complet d'utilisation de serveurs avec Amazon S3, consultez [S3 BucketOps .java](#) dans le référentiel d'exemples de AWS code.

Utiliser la programmation asynchrone

AWS SDK for Java 2.x Il propose des clients asynchrones avec un support d'E/S non bloquant qui implémentent une simultanéité élevée sur quelques threads. Cependant, l'absence totale de blocage des E/S n'est pas garantie. Le client asynchrone peut bloquer les appels dans certains cas, tels que la récupération des informations d'identification, la signature des demandes à l'aide de [AWS Signature Version 4 \(SigV4\)](#) ou la découverte des terminaux.

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. Les méthodes pour le client asynchrone dans la version 2.x des `CompletableFuture` objets de AWS SDK for Java retour qui vous permettent d'accéder à la réponse lorsqu'elle est prête.

Opérations autres que le streaming

Pour les opérations autres que de streaming, les appels de méthodes asynchrones sont similaires à ceux des méthodes synchrones. Cependant, les méthodes asynchrones AWS SDK for Java renvoient un [CompletableFuture](#) objet contenant les résultats de l'opération asynchrone à venir.

Appelez la `CompletableFuture` `whenComplete()` méthode avec une action à terminer lorsque le résultat est disponible. `CompletableFuture` implémente l'`Future` interface, de sorte que vous pouvez également obtenir l'objet de réponse en appelant la `get()` méthode.

Voici un exemple d'opération asynchrone qui appelle une Amazon DynamoDB fonction pour obtenir une liste de tables et reçoit un fichier `CompletableFuture` pouvant contenir un [ListTablesResponse](#) objet. L'action définie dans l'appel de `whenComplete()` est uniquement effectuée lorsque l'appel asynchrone est terminé.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

Code

```
public class DynamoDBAsyncListTables {

    public static void main(String[] args) throws InterruptedException {

        // Create the DynamoDbAsyncClient object
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient client = DynamoDbAsyncClient.builder()
            .region(region)
            .build();

        listTables(client);
    }

    public static void listTables(DynamoDbAsyncClient client) {

        CompletableFuture<ListTablesResponse> response =
client.listTables(ListTablesRequest.builder()
            .build());

        // Map the response to another CompletableFuture containing just the table
names
        CompletableFuture<List<String>> tableNames =
response.thenApply(ListTablesResponse::tableNames);

        // When future is complete (either successfully or in error) handle the
response
        tableNames.whenComplete((tables, err) -> {
            try {
                if (tables != null) {
                    tables.forEach(System.out::println);
                } else {
                    // Handle error
                    err.printStackTrace();
                }
            } finally {
                // Lets the application shut down. Only close the client when you are
completely done with it.
            }
        });
    }
}
```



```
        client.close();
    }
});
tableNames.join();
}
}
```

L'exemple de code suivant montre comment extraire un élément d'une table à l'aide du client asynchrone. Invoquez la `getItem` méthode du `DynamoDbAsyncClient` et transmettez-lui un [GetItemRequest](#) objet avec le nom de la table et la valeur de la clé primaire de l'élément souhaité. C'est généralement ainsi que vous transmettez les données requises par l'opération. Dans cet exemple, notez qu'une valeur `String` est passée.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String
key, String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {

        // Create a GetItemRequest instance
        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName(tableName)
```

```
        .build();

        // Invoke the DynamoDbAsyncClient object's getItem
        java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

        // Convert Set to Map
        Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
        Set<String> keys = map.keySet();
        for (String sinKey : keys) {
            System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Opérations de streaming

Pour les opérations de streaming, vous devez fournir un [AsyncRequestBody](#) pour fournir le contenu de manière incrémentielle, ou un [AsyncResponseTransformer](#) pour recevoir et traiter la réponse.

L'exemple suivant télécharge un fichier de Amazon S3 manière asynchrone à l'aide de l'opération.

PutObject

Importations

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

Code

```
/**
```

```
* To run this AWS code example, ensure that you have setup your development
environment, including your AWS credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
```

```
public class S3AsyncOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "    S3AsyncOps <bucketName> <key> <path>\n\n" +
            "Where:\n" +
            "    bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "    key - the name of the object (for example, book.pdf). \n" +
            "    path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        // Put the object into the bucket
        CompletableFuture<PutObjectResponse> future = client.putObject(objectRequest,
            AsyncRequestBody.fromFile(Paths.get(path)))
```

```
    );
    future.whenComplete((resp, err) -> {
        try {
            if (resp != null) {
                System.out.println("Object uploaded. Details: " + resp);
            } else {
                // Handle error
                err.printStackTrace();
            }
        } finally {
            // Only close the client when you are completely done with it
            client.close();
        }
    });

    future.join();
}
}
```

L'exemple suivant extrait un fichier de Amazon S3 manière asynchrone à l'aide de l'GetObjectopération.

Importations

```
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

Code

```
/**
 * To run this AWS code example, ensure that you have setup your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class S3AsyncStreamOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  S3AsyncStreamOps <bucketName> <objectKey> <path>\n\n" +
            "Where:\n" +
            "  bucketName - the name of the Amazon S3 bucket (for example,  
bucket1). \n\n" +
            "  objectKey - the name of the object (for example, book.pdf). \n" +
            "  path - the local path to the file (for example, C:/AWS/book.pdf).  
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        CompletableFuture<GetObjectResponse> futureGet =
            client.getObject(objectRequest,
                AsyncResponseTransformerToFile(Path.get(path)));

        futureGet.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object downloaded. Details: "+resp);
                } else {
                    err.printStackTrace();
                }
            }
        })
    }
}
```

```
        } finally {
            // Only close the client when you are completely done with it
            client.close();
        }
    });
    futureGet.join();
}
}
```

Opérations avancées

La version AWS SDK for Java 2.x utilise [Netty](#), un framework d'applications réseau asynchrone piloté par des événements, pour gérer les threads d'E/S. Le AWS SDK for Java 2.x crée un `ExecutorService` Behind Netty, pour compléter les futures renvoyés par la demande du client HTTP jusqu'au client Netty. Cette abstraction réduit le risque qu'une application interrompe le processus asynchrone si les développeurs choisissent d'arrêter ou de mettre des threads en veille. Par défaut, chaque client asynchrone crée un pool de threads basé sur le nombre de processeurs et gère les tâches d'une file d'attente au sein du `ExecutorService`.

Les utilisateurs avancés peuvent spécifier la taille de leur groupe de threads quand ils créent un client asynchrone à l'aide de l'option suivante lors de la génération.

Code

```
S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            Executors.newFixedThreadPool(10))
    )
    .build();
```

Pour optimiser les performances, vous pouvez gérer votre propre exécuteur de groupe de threads et l'inclure lors de la configuration de votre client.

```
ThreadPoolExecutor executor = new ThreadPoolExecutor(50, 50,
    10, TimeUnit.SECONDS,
    new LinkedBlockingQueue<>(<custom_value>),
    new ThreadFactoryBuilder()
```

```
        .threadNamePrefix("sdk-async-response").build());

// Allow idle core threads to time out
executor.allowCoreThreadTimeOut(true);

S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            executor
        )
    )
    .build();
```

Travaillez avec HTTP/2 dans AWS SDK for Java

HTTP/2 est une révision majeure du protocole HTTP. Cette nouvelle version offre plusieurs améliorations pour améliorer les performances :

- L'encodage des données binaires fournit un transfert de données plus efficace.
- La compression d'en-tête réduit le nombre d'octets téléchargés par le client, ce qui permet de fournir plus rapidement le contenu au client. Cela s'avère particulièrement utile pour les clients mobiles qui sont déjà limités en bande passante.
- La communication asynchrone bidirectionnelle (multiplexage) permet d'envoyer plusieurs demandes et messages de réponse entre le client et AWS d'être envoyés en même temps sur une seule connexion, plutôt que sur plusieurs connexions, ce qui améliore les performances.

Une mise à niveau de développeurs vers les derniers kits SDK utilise automatiquement HTTP/2 lorsque ce protocole est pris en charge par le service utilisé. De nouvelles interfaces de programmation tirent parti de manière transparente des avantages des fonctions HTTP/2 et offrent de nouvelles façons de générer des applications.

La version AWS SDK for Java 2.x propose de nouvelles API pour le streaming d'événements qui implémentent le protocole HTTP/2. Pour des exemples d'utilisation de ces nouvelles API, consultez la section [Utilisation de Kinesis](#).

Utilisez les métriques du SDK issues du AWS SDK for Java

Avec la AWS SDK for Java version 2.x, vous pouvez collecter des métriques sur les clients du service dans votre application, analyser les résultats Amazon CloudWatch, puis agir en conséquence.

Par défaut, la collecte de métriques est désactivée dans le SDK. Cette rubrique vous aide à l'activer et à le configurer.

Prérequis

Avant de pouvoir activer et utiliser les métriques, vous devez suivre les étapes suivantes :

- Suivez les étapes de [Configuration](#).
- Configurez les dépendances de votre projet (par exemple, dans votre `build.gradle` fichier `pom.xml` ou dans votre fichier) pour utiliser la version `2.14.0` ou une version ultérieure du AWS SDK for Java.

Pour permettre la publication de métriques sur CloudWatch, incluez également `cloudwatch-metric-publisher` avec le `2.14.0` numéro de version ou une version ultérieure dans les dépendances de votre projet.

Par exemple :

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>cloudwatch-metric-publisher</artifactId>
      <version>2.14.0</version>
    </dependency>
  </dependencies>
</project>
```



```
</dependencies>
</project>
```

- Activez `cloudwatch:PutMetricData` les autorisations pour l'identité IAM utilisée par l'éditeur de métriques afin de permettre au SDK for Java d'écrire des métriques.

Comment activer la collecte de métriques

Vous pouvez activer les métriques dans votre application pour un client de service ou sur des demandes individuelles.

Activer les métriques pour une demande spécifique

La classe suivante montre comment activer l'éditeur de CloudWatch métriques pour une demande adressée à Amazon DynamoDB. Il utilise la configuration par défaut de l'éditeur de métriques.

```
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;

public class DefaultConfigForRequest {
    // Use one MetricPublisher for your application. It can be used with requests or
    // service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.create();
        // Publish metrics the for ListTables operation.
        ddb.listTables(ListTablesRequest.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build());

        // Perform more work in your application.

        // A MetricsPublisher has its own lifecycle independent of any service client
        // or request that uses it.
        // If you no longer need the publisher, close it to free up resources.
        metricsPub.close(); // All metrics stored in memory are flushed to CloudWatch.

        // Perform more work with the DynamoDbClient instance without publishing
        // metrics.
    }
}
```

```
        // Close the service client when you no longer need it.
        ddb.close();
    }
}
```

Important

Assurez-vous que votre application appelle `close` l'[MetricPublisher](#) instance lorsque le client de service n'est plus utilisé. Si vous ne le faites pas, cela peut entraîner des fuites de thread ou de descripteur de fichier.

Activer les métriques récapitulatives pour un client de service spécifique

L'extrait de code suivant montre comment activer un éditeur de CloudWatch mesures avec des paramètres par défaut pour un client de service.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

Personnaliser l'éditeur de métriques

La classe suivante montre comment configurer une configuration personnalisée pour l'éditeur de métriques pour un client de service spécifique. Les personnalisations incluent le chargement d'un profil spécifique, la spécification d'une AWS région dans laquelle l'éditeur de métriques envoie des demandes et la personnalisation de la fréquence à laquelle l'éditeur envoie des métriques.

CloudWatch

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.metrics.CoreMetric;
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import java.time.Duration;
```

```
public class CustomConfigForDDBClient {
    // Use one MetricPublisher for your application. It can be used with requests or
    // service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
        .cloudWatchClient(CloudWatchAsyncClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
            .build())
        .uploadFrequency(Duration.ofMinutes(5))
        .maximumCallsPerUpload(100)
        .namespace("ExampleSDKV2Metrics")
        .detailedMetrics(CoreMetric.API_CALL_DURATION)
        .build();

    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build();
        // Publish metrics for DynamoDB operations.
        ddb.listTables();
        ddb.describeEndpoints();
        ddb.describeLimits();
        // Perform more work in your application.

        // A MetricsPublisher has its own lifecycle independent of any service client
        // or request that uses it.
        // If you no longer need the publisher, close it to free up resources.
        metricsPub.close(); // All metrics stored in memory are flushed to CloudWatch.

        // Perform more work with the DynamoDbClient instance without publishing
        // metrics.
        // Close the service client when you no longer need it.
        ddb.close();
    }
}
```

Les personnalisations présentées dans l'extrait de code précédent ont les effets suivants.

- La `cloudWatchClient` méthode vous permet de personnaliser le CloudWatch client utilisé pour envoyer les métriques. Dans cet exemple, nous utilisons une région différente de la région par défaut `us-east-1` où le client envoie des métriques. Nous utilisons également un profil nommé

différent, `cloudwatch`, dont les informations d'identification seront utilisées pour authentifier les demandes adressées à `CloudWatch`. Ces informations d'identification doivent être autorisées à `cloudwatch:PutMetricData`.

- La `uploadFrequency` méthode vous permet de spécifier la fréquence à laquelle l'éditeur télécharge les `CloudWatch` métriques. La valeur par défaut est une fois par minute.
- La `maximumCallsPerUpload` méthode limite le nombre d'appels effectués par téléchargement. La valeur par défaut est illimitée.
- Par défaut, le SDK pour Java 2.x publie les métriques sous l'espace de noms `AwsSdk/JavaSdk2`. Vous pouvez utiliser `namespace` cette méthode pour spécifier une valeur différente.
- Par défaut, le SDK publie des métriques récapitulatives. Les mesures récapitulatives comprennent la moyenne, le minimum, le maximum, la somme et le nombre d'échantillons. En spécifiant une ou plusieurs métriques du SDK dans la `detailedMetrics` méthode, le SDK publie des données supplémentaires pour chaque métrique. Ces données supplémentaires permettent d'obtenir des statistiques percentiles, telles que `p90` et `p99`, que vous pouvez interroger. `CloudWatch` Les métriques détaillées sont particulièrement utiles pour les métriques de `latencyAPICallDuration`, telles que celles qui mesurent la end-to-end latence pour les demandes des clients du SDK. Vous pouvez utiliser les champs de la [CoreMetric](#) classe pour spécifier d'autres métriques courantes du SDK.

Quand les statistiques sont-elles disponibles ?

Les métriques sont généralement disponibles dans les 5 à 10 minutes suivant leur émission par le SDK for Java. Pour des up-to-date statistiques précises, vérifiez `Cloudwatch` au moins 10 minutes après avoir émis les métriques depuis vos applications Java.

Quelles sont les informations collectées ?

La collecte de métriques inclut les éléments suivants :

- Nombre de demandes d'API, y compris si elles aboutissent ou échouent
- Informations sur les AWS services que vous appelez dans vos demandes d'API, y compris les exceptions renvoyées
- Durée des différentes opérations telles que le marshalling, la signature et les requêtes HTTP
- Mesures du client HTTP, telles que le nombre de connexions ouvertes, le nombre de demandes en attente et le nom du client HTTP utilisé

Note

Les métriques disponibles varient en fonction du client HTTP.

Pour une liste complète, consultez la section [Mesures relatives aux clients du service](#).

Comment puis-je utiliser ces informations ?

Vous pouvez utiliser les métriques collectées par le SDK pour surveiller les clients de service de votre application. Vous pouvez examiner les tendances générales d'utilisation, identifier les anomalies, examiner les exceptions renvoyées par les clients du service ou approfondir la compréhension d'un problème particulier. En utilisant Amazon CloudWatch, vous pouvez également créer des alarmes pour vous avertir dès que votre application atteint une condition que vous définissez.

Pour plus d'informations, consultez les sections [Utilisation Amazon CloudWatch des métriques](#) et [Utilisation des Amazon CloudWatch alarmes](#) dans le [guide de Amazon CloudWatch l'utilisateur](#).

Mesures relatives aux clients du service

Avec le AWS SDK for Java 2.x, vous pouvez collecter des statistiques auprès des clients du service dans votre application, puis publier (générer) ces statistiques [sur Amazon CloudWatch](#).

Ces tableaux répertorient les métriques que vous pouvez collecter et les exigences relatives à l'utilisation du client HTTP.

Pour plus d'informations sur l'activation et la configuration des métriques pour le SDK, consultez la section [Activation des métriques du SDK](#).

Métriques collectées à chaque demande

Nom des métriques	Description	Type
ApiCallDuration	Le temps total nécessaire pour terminer une demande (toutes les nouvelles tentatives incluses).	Durée
ApiCallSuccessful	Vrai si l'appel d'API a réussi, faux dans le cas contraire.	Booléen

Nom des métriques	Description	Type
CredentialsFetchDuration	Le temps nécessaire pour récupérer les informations de AWS signature de la demande.	Durée
EndpointResolveDuration	Le temps nécessaire pour résoudre le point de terminais on utilisé pour l'appel d'API.	Durée
MarshallingDuration	Le temps nécessaire pour transformer une requête du SDK en une requête HTTP.	Durée
OperationName	Le nom de l' AWS API à laquelle la demande est envoyée.	Chaîne
RetryCount	Nombre de fois que le SDK a retenté l'appel d'API.	Entier
ServiceId	ID de service du pour Service AWS lequel la demande d'API est effectuée.	Chaîne
TokenFetchDuration	Le temps nécessaire pour récupérer les identifiants de signature du jeton pour la demande.	Durée

Mesures collectées pour chaque tentative de demande

Chaque appel d'API peut nécessiter plusieurs tentatives avant de recevoir une réponse. Ces statistiques sont collectées pour chaque tentative.

Indicateurs de base

Nom des métriques	Description	Type
AwsExtendedRequestId	L'ID de demande étendu de la demande de service.	Chaîne
AwsRequestId	ID de demande de la demande de service.	Chaîne
BackoffDelayDuration	Durée pendant laquelle le SDK a attendu avant cette tentative d'appel d'API.	Durée
ErrorType	Type d'erreur survenue lors d'une tentative d'appel.	Chaîne
ReadThroughput	Débit de lecture du client.	Double
ServiceCallDuration	Le temps nécessaire pour se connecter au service, envoyer la demande et recevoir le code d'état HTTP et l'en-tête de la réponse.	Durée
SigningDuration	Le temps nécessaire pour signer la requête HTTP.	Durée
TimeToFirstByte	Temps écoulé entre l'envoi de la requête HTTP (y compris l'acquisition d'une connexion) et la réception du premier octet des en-têtes de la réponse.	Durée
TimeToLastByte	Temps écoulé entre l'envoi de la requête HTTP (y compris l'acquisition d'une connexion)	Durée

Nom des métriques	Description	Type
	et la réception du dernier octet de la réponse.	
UnmarshallingDuration	Le temps nécessaire pour désamorcer une réponse HTTP à une réponse du SDK.	Durée

Métriques HTTP

Nom des métriques	Description	Type	Client HTTP requis*
AvailableConcurrency	Le nombre de demandes simultanées restantes qui peuvent être prises en charge par le client HTTP sans qu'il soit nécessaire d'établir une autre connexion.	Entier	Apache, Netty, CRT
ConcurrencyAcquireDuration	Le temps nécessaire pour acquérir un canal à partir du pool de connexions.	Durée	Apache, Netty, CRT
HttpClientName	Nom du protocole HTTP utilisé pour la demande.	Chaîne	Apache, Netty, CRT
HttpStatuscode	Le code d'état renvoyé avec la réponse HTTP.	Entier	N'importe quel compte
LeasedConcurrency	Le nombre de requêtes en cours	Entier	Apache, Netty, CRT

Nom des métriques	Description	Type	Client HTTP requis*
	d'exécution par le client HTTP.		
LocalStreamWindowSize	Taille de la fenêtre HTTP/2 locale en octets pour le flux sur lequel cette demande a été exécutée.	Entier	Netty
MaxConcurrency	Le nombre maximal de demandes simultanées prises en charge par le client HTTP.	Entier	Apache, Netty, CRT
PendingConcurrency Acquires	Nombre de demandes bloquées, en attente de la disponibilité d'une autre connexion TCP ou d'un nouveau flux depuis le pool de connexions.	Entier	Apache, Netty, CRT
RemoteStreamWindowSize	Taille de la fenêtre HTTP/2 distante en octets pour le flux sur lequel cette demande a été exécutée.	Entier	Netty

Les termes utilisés dans la colonne signifient :

- Apache : le client HTTP basé sur Apache () [ApacheHttpClient](#)
- Netty : le client HTTP basé sur Netty () [NettyNioAsyncHttpClient](#)
- CRT : le client HTTP AWS basé sur CRT () [AwsCrtAsyncHttpClient](#)

- N'importe lequel : la collecte de données métriques ne dépend pas du client HTTP ; cela inclut le client HTTP basé sur URLConnection () [URLConnectionHttpClient](#)

Travaillez en Services AWS utilisant le AWS SDK for Java 2.x

Cette section fournit de courts didacticiels et des conseils sur la façon d'utiliser Select Services AWS. Pour un ensemble complet d'exemples, consultez la [section Exemples de code](#).

Rubriques

- [Travailler avec CloudWatch](#)
- [AWS services de base de données et AWS SDK for Java 2.x](#)
- [Travaillez avec DynamoDB](#)
- [Travaillez avec Amazon EC2](#)
- [Travailler avec IAM](#)
- [Travaillez avec Kinesis](#)
- [AWS Lambda Fonctions d'appel, de liste et de suppression](#)
- [Travaillez avec Amazon S3](#)
- [Travailler avec Amazon Simple Notification Service](#)
- [Travaillez avec Amazon Simple Queue Service](#)
- [Travailler avec Amazon Transcribe](#)

Travailler avec CloudWatch

Cette section fournit des exemples de programmation d'[Amazon](#) à l'aide CloudWatch de la version AWS SDK for Java 2.x.

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

Les exemples suivants 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

- [Obtenez des statistiques auprès de CloudWatch](#)
- [Publiez des données métriques personnalisées sur CloudWatch](#)
- [Travailler avec des CloudWatch alarmes](#)
- [Utiliser Amazon CloudWatch Events](#)

Obtenez des statistiques auprès de CloudWatch

Affichage des métriques

Pour répertorier CloudWatch les métriques, créez une méthode [ListMetricsRequest](#) et appelez CloudWatchClient la `listMetrics` méthode. Vous pouvez utiliser `ListMetricsRequest` pour filtrer les métriques renvoyées par espace de noms, nom de métrique ou dimension.

Note

Une liste des mesures et des dimensions publiées par les AWS services se trouve dans la [référence Amazon CloudWatch des mesures et dimensions](#) du guide de l'Amazon CloudWatch utilisateur.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
```

Code

```
public static void listMets( CloudWatchClient cw, String namespace) {

    boolean done = false;
    String nextToken = null;

    try {
```

```
while(!done) {

    ListMetricsResponse response;

    if (nextToken == null) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        response = cw.listMetrics(request);
    } else {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .nextToken(nextToken)
            .build();

        response = cw.listMetrics(request);
    }

    for (Metric metric : response.metrics()) {
        System.out.printf(
            "Retrieved metric %s", metric.metricName());
        System.out.println();
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        nextToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Les métriques sont renvoyées dans un [ListMetricsResponse](#) en appelant sa `getMetrics` méthode.

Les résultats peuvent être paginés. Pour extraire le prochain lot de résultats, appelez `nextToken` sur l'objet réponse et utilisez la valeur du jeton pour générer un objet nouvelle demande. Puis, appelez à nouveau la méthode `listMetrics` avec la nouvelle demande.

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

En savoir plus

- [ListMetrics](#) dans la référence de Amazon CloudWatch l'API

Publiez des données métriques personnalisées sur CloudWatch

Un certain nombre de AWS services publient [leurs propres métriques dans des](#) espaces de noms commençant par « AWS ». Vous pouvez également publier des données métriques personnalisées en utilisant votre propre espace de noms (à condition qu'il ne commence pas par AWS « »).

Publier des données de métriques personnalisées

Pour publier vos propres données métriques, appelez la `putMetricData` méthode `CloudWatchClient`'s avec un [PutMetricDataRequest](#). Ils `PutMetricDataRequest` doivent inclure l'espace de noms personnalisé à utiliser pour les données, ainsi que des informations sur le point de données lui-même dans un [MetricDatum](#) objet.

Note

Vous ne pouvez pas spécifier un espace de noms commençant par « AWS ». Les espaces de noms commençant par AWS « » sont réservés aux Amazon Web Services produits.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
```

Code

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {

    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object
        String time =
            ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension).build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum).build();

        cw.putMetricData(request);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.printf("Successfully put data point %f", dataPoint);
}
```

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

En savoir plus

- [Utilisez Amazon CloudWatch les métriques](#) dans le guide de Amazon CloudWatch l'utilisateur.
- [AWSEspaces de noms](#) dans le guide de Amazon CloudWatch l'utilisateur.
- [PutMetricData](#) dans la référence de Amazon CloudWatch l'API.

Travailler avec des CloudWatch alarmes

Créer une alarme

Pour créer une alarme basée sur une CloudWatch métrique, appelez la `putMetricAlarm` méthode `CloudWatchClient`'s avec un [PutMetricAlarmRequest](#) rempli des conditions d'alarme.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

Code

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

    try {
        Dimension dimension = Dimension.builder()
            .name("InstanceId")
            .value(instanceId).build();

        PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
            .alarmName(alarmName)
            .comparisonOperator(
                ComparisonOperator.GREATER_THAN_THRESHOLD)
            .evaluationPeriods(1)
            .metricName("CPUUtilization")
            .namespace("AWS/EC2")
            .period(60)
            .statistic(Statistic.AVERAGE)
            .threshold(70.0)
            .actionsEnabled(false)
            .alarmDescription(
                "Alarm when server CPU utilization exceeds 70%")
            .unit(StandardUnit.SECONDS)
            .dimensions(dimension)
```



```
        .build();

        cw.putMetricAlarm(request);
        System.out.printf(
            "Successfully created alarm with name %s", alarmName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Répertorier les alarmes

Pour répertorier les CloudWatch alarmes que vous avez créées, appelez la `describeAlarms` méthode `CloudWatchClient`'s avec un [DescribeAlarmsRequest](#) que vous pouvez utiliser pour définir les options du résultat.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
```

Code

```
public static void desCWAAlarms( CloudWatchClient cw) {

    try {

        boolean done = false;
        String newToken = null;

        while(!done) {
            DescribeAlarmsResponse response;

            if (newToken == null) {
```

```
        DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
        response = cw.describeAlarms(request);
    } else {
        DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
            .nextToken(newToken)
            .build();
        response = cw.describeAlarms(request);
    }

    for(MetricAlarm alarm : response.metricAlarms()) {
        System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
```

La liste des alarmes peut être obtenue `MetricAlarms` en appelant [DescribeAlarmsResponse](#) le code renvoyé par `describeAlarms`.

Les résultats peuvent être paginés. Pour extraire le prochain lot de résultats, appelez `nextToken` sur l'objet réponse et utilisez la valeur du jeton pour générer un objet nouvelle demande. Puis, appelez à nouveau la méthode `describeAlarms` avec la nouvelle demande.

Note

Vous pouvez également récupérer les alarmes pour une métrique spécifique en utilisant la `describeAlarmsForMetric` méthode `CloudWatchClient`'s. Son utilisation est similaire à `describeAlarms`.

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

Supprimer des alertes

Pour supprimer des CloudWatch alarmes, appelez la `deleteAlarms` méthode `CloudWatchClient`'s [DeleteAlarmsRequest](#) en indiquant un ou plusieurs noms d'alarmes que vous souhaitez supprimer.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
```

Code

```
public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {

    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

En savoir plus

- [Utilisation des Amazon CloudWatch alarmes](#) dans le guide de Amazon CloudWatch l'utilisateur
- [PutMetricAlarm](#) dans la référence de Amazon CloudWatch l'API
- [DescribeAlarms](#) dans la référence de Amazon CloudWatch l'API
- [DeleteAlarms](#) dans la référence de Amazon CloudWatch l'API

Utiliser Amazon CloudWatch Events

CloudWatch Events fournit un flux d'événements système en temps quasi réel décrivant les modifications apportées aux AWS ressources des Amazon EC2 instances, des Lambda fonctions, des Kinesis flux, Amazon ECS des tâches, des machines d' Step Functions état, des Amazon SNS sujets, des Amazon SQS files d'attente ou des 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.

Amazon EventBridge est l'[évolution](#) des CloudWatch événements. Les deux services utilisent la même API. Vous pouvez donc continuer à utiliser le [client CloudWatch Events](#) fourni par le SDK ou migrer vers le [EventBridge client](#) du SDK pour Java pour la fonctionnalité CloudWatch Events. CloudWatch [La documentation du Guide de l'utilisateur](#) des événements et les [références des API](#) sont désormais disponibles sur les sites de EventBridge documentation.

Ajouter des événements

Pour ajouter CloudWatch des événements personnalisés, appelez la `CloudWatchEventsClient`'s `putEvents` méthode avec un [PutEventsRequest](#) objet contenant un ou plusieurs [PutEventsRequestEntry](#) objets fournissant 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 software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;
```

Code

```
public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {
```

```
try {

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

    PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
        .detail(EVENT_DETAILS)
        .detailType("sampleSubmitted")
        .resources(resourceArn)
        .source("aws-sdk-java-cloudwatch-example")
        .build();

    PutEventsRequest request = PutEventsRequest.builder()
        .entries(requestEntry)
        .build();

    cwe.putEvents(request);
    System.out.println("Successfully put CloudWatch event");

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

Ajouter des règles

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

Importations

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

Code

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {

    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            roleArn, response.ruleArn());
    } catch (
        CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Ajouter des cibles

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

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

Importations

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsResponse;
```

```
import software.amazon.awssdk.services.cloudwatchevents.model.Target;
```

Code

```
public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {

    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        PutTargetsResponse response = cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

En savoir plus

- [Ajouter des événements PutEvents](#) dans le guide de EventBridge l'utilisateur Amazon
- [Expressions de planification pour les règles](#) dans le guide de EventBridge l'utilisateur Amazon
- [Types d'événements figurant CloudWatch Events](#) dans le guide de EventBridge l'utilisateur Amazon
- [Modèles d'événements](#) présentés dans le guide de EventBridge l'utilisateur Amazon
- [PutEvents](#) dans le Amazon EventBridge API Reference
- [PutTargets](#) dans le Amazon EventBridge API Reference
- [PutRule](#) dans le Amazon EventBridge API Reference

AWS services de base de données et AWS SDK for Java 2.x

AWS [propose plusieurs types de bases de données : relationnelle, clé-valeur, en mémoire, documentaire, etc.](#) La prise en charge du SDK for Java 2.x varie en fonction de la nature du service de base de données dans AWS.

Certains services de base de données, par exemple le service [Amazon DynamoDB](#), disposent d'API de service Web pour gérer AWS la ressource (base de données) ainsi que d'API de service Web pour interagir avec les données. [Dans le SDK pour Java 2.x, ces types de services ont des clients de service dédiés, par exemple DynamoDBClient.](#)

D'autres services de base de données disposent d'API de service Web qui interagissent avec la ressource, comme l'API [Amazon DocumentDB](#) (pour la gestion des clusters, des instances et des ressources), mais n'ont pas d'API de service Web pour travailler avec les données. Le SDK pour Java 2.x possède une interface [DocDbClient](#) correspondante pour travailler avec la ressource. Cependant, vous avez besoin d'une autre API Java, telle que [MongoDB pour Java, pour](#) travailler avec les données.

Utilisez les exemples ci-dessous pour savoir comment utiliser le SDK pour les clients de service Java 2.x avec les différents types de bases de données.

Exemples d'Amazon DynamoDB

Travailler avec les données

Client du service SDK : [DynamoDbClient](#)

Exemple : application [REST React/Spring utilisant DynamoDB](#)

Exemples : [plusieurs exemples de DynamoDB](#)

Client du service SDK : [DynamoDbEnhancedClient](#)

Exemple : application [REST React/Spring utilisant DynamoDB](#)

Utilisation de la base de données

Client du service SDK : [DynamoDbClient](#)

Exemples : [CreateTable, ListTables, DeleteTable](#)

Travailler avec les données

Utilisation de la base de données

Exemples : [plusieurs exemples DynamoDB](#)
([noms commençant par](#) « Enhanced »)

Consultez [d'autres exemples de DynamoDB](#) dans la section des exemples de code guidés de ce guide.

Exemples Amazon RDS

Travailler avec les données	Utilisation de la base de données
API non SDK : JDBC, version SQL spécifique à la base de données ; votre code gère les connexions à la base de données ou un pool de connexions.	Client du service SDK : RdsClient
Exemple : application REST React/Spring utilisant MySQL	Exemples : Plusieurs RdsClient exemples

Exemples d'Amazon Redshift

Travailler avec les données	Utilisation de la base de données
Client du service SDK : RedshiftDataClient	Client du service SDK : RedshiftClient
Exemples : Plusieurs RedshiftDataClient exemples	Exemples : Plusieurs RedshiftClient exemples
Exemple : application REST React/Spring utilisant RedshiftDataClient	

Exemples d'Amazon Aurora Serverless v2

Travailler avec les données	Utilisation de la base de données
Client du service SDK : RdsDataClient	Client du service SDK : RdsClient
Exemple : application REST React/Spring utilisant RdsDataClient	Exemples : Plusieurs RdsClient exemples

Exemples d'Amazon DocumentDB

Travailler avec les données	Utilisation de la base de données
API non SDK : bibliothèque Java spécifique à MongoDB (par exemple MongoDB pour Java) ; votre code gère les connexions à la base de données ou un pool de connexions.	Client du service SDK : DocDbClient
Exemples : Guide du développeur de DocumentDB (Mongo) (sélectionnez l'onglet « Java »)	

Travaillez avec DynamoDB

Cette section fournit des exemples qui vous montrent comment utiliser [DynamoDB](#).

Les exemples suivants utilisent le client DynamoDB standard de bas niveau [DynamoDbClient](#) () de la version 2.x. AWS SDK for Java

- [the section called “Travaillez avec des tables dans DynamoDB”](#)
- [the section called “Travaillez avec des objets dans DynamoDB”](#)

Le SDK propose également le client [amélioré DynamoDB](#) qui fournit une approche orientée objet de haut niveau pour travailler avec DynamoDB. La section suivante décrit ce client en détail.

- [the section called “ Associer des objets à des éléments DynamoDB”](#)

Travaillez avec des tables dans DynamoDB

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

Pour chaque table, vous devez définir :

- Nom de table unique pour votre compte et votre 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 du Amazon DynamoDB développeur.

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

Note

Amazon DynamoDB la [tarification](#) est basée sur les valeurs de débit provisionnées que vous définissez sur 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.

Créer une table

Utilisez `DynamoDbClient`'s `createTable` cette méthode pour créer une nouvelle DynamoDB 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.

Note

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

Créer une table avec une clé primaire simple

Ce code crée une table avec un attribut qui est la clé primaire simple de la table. L'exemple utilise [AttributeDefinition](#) et [KeySchemaElement](#) objet pour [CreateTableRequest](#)

Importations

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

Code

```
public static String createTable(DynamoDbClient ddb, String tableName, String key)
{
    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
```

```
        .keyType(KeyType.HASH)
        .build()
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .tableName(tableName)
    .build();

String newTable = "";
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created
    WaiterResponse<DescribeTableResponse> waiterResponse =
ddbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);

    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

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

Créer une table avec une clé primaire composite

L'exemple suivant crée une table avec deux attributs. Les deux attributs sont utilisés pour la clé primaire composite.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
```

```
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

Code

```
public static String createTableComKey(DynamoDbClient ddb, String tableName) {
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(
            AttributeDefinition.builder()
                .attributeName("Language")
                .attributeType(ScalarAttributeType.S)
                .build(),
            AttributeDefinition.builder()
                .attributeName("Greeting")
                .attributeType(ScalarAttributeType.S)
                .build())
        .keySchema(
            KeySchemaElement.builder()
                .attributeName("Language")
                .keyType(KeyType.HASH)
                .build(),
            KeySchemaElement.builder()
                .attributeName("Greeting")
                .keyType(KeyType.RANGE)
                .build())
        .provisionedThroughput(
            ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10)).build())
        .tableName(tableName)
        .build();

    String tableId = "";

    try {
        CreateTableResponse result = ddb.createTable(request);
        tableId = result.tableDescription().tableId();
        return tableId;
    }
```

```
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

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

Répertoire des tables

Vous pouvez répertorier les tables d'une région donnée en appelant la `DynamoDbClient`'s `listTables` méthode.

Note

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

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.List;
```

Code

```
public static void listAllTables(DynamoDbClient ddb){

    boolean moreTables = true;
    String lastName = null;

    while(moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
```

```
        ListTablesRequest request = ListTablesRequest.builder().build();
        response = ddb.listTables(request);
    } else {
        ListTablesRequest request = ListTablesRequest.builder()
            .exclusiveStartTableName(lastName).build();
        response = ddb.listTables(request);
    }

    List<String> tableNames = response.tableNames();

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

    lastName = response.lastEvaluatedTableName();
    if (lastName == null) {
        moreTables = false;
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
System.out.println("\nDone!");
}
```

Par défaut, jusqu'à 100 tables sont renvoyées par appel. À utiliser `lastEvaluatedTableName` sur l'[ListTablesResponse](#) 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.

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

Décrire (obtenir des informations) sur une table

Utilisez `DynamoDbClient`'s `describeTable` cette méthode pour obtenir des informations sur une table.

Note

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

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```

Code

```
public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName ) {

    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo =
            ddb.describeTable(request).table();

        if (tableInfo != null) {
            System.out.format("Table name   : %s\n",
                tableInfo.tableName());
            System.out.format("Table ARN   : %s\n",
                tableInfo.tableArn());
            System.out.format("Status      : %s\n",
                tableInfo.tableStatus());
            System.out.format("Item count  : %d\n",
                tableInfo.itemCount().longValue());
            System.out.format("Size (bytes): %d\n",
                tableInfo.tableSizeBytes().longValue());

            ProvisionedThroughputDescription throughputInfo =
```

```
        tableInfo.provisionedThroughput();
        System.out.println("Throughput");
        System.out.format("  Read Capacity : %d\n",
            throughputInfo.readCapacityUnits().longValue());
        System.out.format("  Write Capacity: %d\n",
            throughputInfo.writeCapacityUnits().longValue());

        List<AttributeDefinition> attributes =
            tableInfo.attributeDefinitions();
        System.out.println("Attributes");

        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n",
                a.attributeName(), a.attributeType());
        }
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("\nDone!");
}
```

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

Modifier (mettre à jour) une table

Vous pouvez modifier les valeurs de débit provisionnées de votre table à tout moment en appelant la `DynamoDbClient`'s `updateTable` méthode.

Note

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

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
public static void updateDynamoDBTable(DynamoDbClient ddb,
                                       String tableName,
                                       Long readCapacity,
                                       Long writeCapacity) {

    System.out.format(
        "Updating %s with new provisioned throughput values\n",
        tableName);
    System.out.format("Read capacity : %d\n", readCapacity);
    System.out.format("Write capacity : %d\n", writeCapacity);

    ProvisionedThroughput tableThroughput = ProvisionedThroughput.builder()
        .readCapacityUnits(readCapacity)
        .writeCapacityUnits(writeCapacity)
        .build();

    UpdateTableRequest request = UpdateTableRequest.builder()
        .provisionedThroughput(tableThroughput)
        .tableName(tableName)
        .build();

    try {
        ddb.updateTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}
```

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

Supprimer une table

Pour supprimer une table, appelez la `DynamoDbClient`'s `deleteTable` méthode et indiquez 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 software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
```

Code

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

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

En savoir plus

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

Travaillez avec des objets 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 plus d'informations, consultez la section [Règles de dénomination et types de données](#) dans le Guide du Amazon DynamoDB développeur.

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

Appelez la `getItem` méthode `DynamoDbClient`'s et transmettez-lui un [GetItemRequest](#) objet avec le nom de la table et la valeur de la clé primaire de l'élément souhaité. Elle renvoie un [GetItemResponse](#) objet avec tous les attributs de cet élément. Vous pouvez spécifier une ou plusieurs [expressions de projection](#) dans `GetItemRequest` pour extraire des attributs spécifiques.

Vous pouvez utiliser la `item()` méthode de l'`GetItemResponse` objet renvoyé pour récupérer une [carte](#) des paires clé (chaîne [AttributeValue](#)) et valeur () associées à l'élément.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
```

Code

```
public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal ) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
```

```
        .build();

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

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", key);
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Extraire (get) un élément d'une table à l'aide du client asynchrone

Invoquez la `getItem` méthode du `DynamoDbAsyncClient` et transmettez-lui un [GetItemRequest](#) objet avec le nom de la table et la valeur de la clé primaire de l'élément souhaité.

Vous pouvez renvoyer une instance [Collection](#) avec tous les attributs de cet élément (reportez-vous à l'exemple suivant).

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String
key, String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {

        // Create a GetItemRequest instance
        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName(tableName)
            .build();

        // Invoke the DynamoDbAsyncClient object's getItem
        java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

        // Convert Set to Map
        Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
        Set<String> keys = map.keySet();
        for (String sinKey : keys) {
            System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Ajouter 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 votre compte et votre région, un [ResourceNotFoundException](#) est généré.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

Code

```
public static void putItemInTable(DynamoDbClient ddb,
                                  String tableName,
                                  String key,
                                  String keyVal,
                                  String albumTitle,
                                  String albumTitleValue,
                                  String awards,
                                  String awardVal,
                                  String songTitle,
                                  String songTitleVal){

    HashMap<String,AttributeValue> itemValues = new
    HashMap<String,AttributeValue>();

    // Add all content to the table
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle,
    AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
```



```
        .build();

    try {
        ddb.putItem(request);
        System.out.println(tableName + " was successfully updated");
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be found.
\n", tableName);
        System.err.println("Be sure that it exists and that you've typed its name
correctly!");
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Mettre à jour un élément existant dans une table

Vous pouvez mettre à jour un attribut pour un élément déjà existant dans une table à l'aide de la méthode `updateItem` d'`DynamoDbClient`, en 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 votre compte et votre région, ou si l'élément identifié par la clé primaire que vous avez transmise n'existe pas, un [ResourceNotFoundException](#) est généré.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;
```

Code

```
public static void updateTableItem(DynamoDbClient ddb,
                                   String tableName,
                                   String key,
                                   String keyVal,
                                   String name,
                                   String updateVal){

    HashMap<String,AttributeValue> itemKey = new HashMap<String,AttributeValue>();

    itemKey.put(key, AttributeValue.builder().s(keyVal).build());

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

    // Update the column specified by name with updatedVal
    updatedValues.put(name, AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s(updateVal).build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}
```

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

Supprimer un élément existant d'une table

Vous pouvez supprimer un élément existant dans une table en utilisant la `deleteItem` méthode `DynamoDbClient`'s et en fournissant un nom de table ainsi que la valeur de la clé primaire.

Note

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

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;
```

Code

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {

    HashMap<String,AttributeValue> keyToGet =
        new HashMap<String,AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();

    try {
```

```
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

En savoir plus

- [Directives relatives à l'utilisation des éléments](#) du guide du Amazon DynamoDB développeur
- [Utilisation des éléments contenus DynamoDB dans](#) le guide du Amazon DynamoDB développeur

Mappez des objets Java à des éléments DynamoDB à l'aide du AWS SDK for Java 2.x

L'API [DynamoDB Enhanced Client](#) est une bibliothèque de haut niveau qui succède à la classe `DynamoDBMapper` du SDK for Java v1.x. Il offre un moyen simple de mapper des classes côté client à des tables DynamoDB. Vous définissez les relations entre les tables et les classes de données correspondantes dans votre code. Après avoir défini ces relations, vous pouvez effectuer de manière intuitive diverses opérations de création, de lecture, de mise à jour ou de suppression (CRUD) sur des tables ou des éléments dans DynamoDB.

L'API DynamoDB Enhanced Client inclut également l'API [Enhanced Document](#) qui vous permet de travailler avec des éléments de type document qui ne suivent pas un schéma défini.

L'API client améliorée DynamoDB est abordée dans les rubriques suivantes.

- [Commencez à utiliser l'API client améliorée DynamoDB](#)
- [Découvrez les principes de base de l'API client améliorée DynamoDB](#)
- [Utiliser les fonctionnalités de cartographie avancées](#)
- [Travaillez avec des documents JSON avec l'API de document améliorée pour DynamoDB](#)
- [Utiliser des extensions](#)
- [Utiliser l'API client améliorée DynamoDB de manière asynchrone](#)
- [Annotations de classes de données](#)

Commencez à utiliser l'API client améliorée DynamoDB

Le didacticiel suivant présente les principes fondamentaux dont vous avez besoin pour utiliser l'API client améliorée DynamoDB.

Ajouter des dépendances

Pour commencer à utiliser l'API client améliorée DynamoDB dans votre projet, ajoutez une dépendance à l'artefact Maven. dynamodb-enhanced Cela est illustré dans les exemples suivants.

Maven

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version><VERSION></version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
  </dependencies>
  ...
</project>
```

Effectuez une recherche dans le référentiel central Maven pour trouver la [dernière version](#) et remplacez-la <VERSION> par cette valeur.

Gradle

```
repositories {
    mavenCentral()
}
dependencies {
```

```
implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
implementation("software.amazon.awssdk:dynamodb-enhanced")
...
}
```

Effectuez une recherche dans le référentiel central Maven pour trouver la [dernière version](#) et remplacez-la <VERSION> par cette valeur.

Générer un **TableSchema** à partir d'une classe de données

A [TableSchema](#) permet au client amélioré de mapper les valeurs d'attribut DynamoDB vers et depuis vos classes côté client. Dans ce didacticiel, vous découvrirez `TableSchema` s dérivé d'une classe de données statique et généré à partir de code à l'aide d'un générateur.

Utiliser une classe de données annotée

Le SDK pour Java 2.x inclut [un ensemble d'annotations que vous pouvez utiliser avec une classe de données](#) pour générer `TableSchema` rapidement un fichier permettant de mapper vos classes à des tables.

Commencez par créer une classe de données conforme à la [JavaBean spécification](#). La spécification exige qu'une classe ait un constructeur public sans argument et dispose de getters et setters pour chaque attribut de la classe. Incluez une annotation au niveau de la classe pour indiquer que la classe de données est une `DynamoDbBean`. Incluez également, au minimum, une `DynamoDbPartitionKey` annotation sur le getter ou le setter pour l'attribut clé primaire.

Vous pouvez appliquer des [annotations au niveau des attributs](#) aux getters ou aux setters, mais pas aux deux.

Note

Le terme `property` est normalement utilisé pour une valeur encapsulée dans un `JavaBean`. Toutefois, ce guide utilise ce terme à la `attribute` place, par souci de cohérence avec la terminologie utilisée par `DynamoDB`.

La `Customer` classe suivante affiche les annotations qui lient la définition de classe à une table `DynamoDB`.

classe **Customer**

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

@DynamoDbBean
public class Customer {

    private String id;
    private String name;
    private String email;
    private Instant regDate;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }

    public void setId(String id) { this.id = id; }

    public String getCustName() { return this.name; }

    public void setCustName(String name) { this.name = name; }

    @DynamoDbSortKey
    public String getEmail() { return this.email; }

    public void setEmail(String email) { this.email = email; }

    public Instant getRegistrationDate() { return this.regDate; }

    public void setRegistrationDate(Instant registrationDate) { this.regDate =
registrationDate; }

    @Override
    public String toString() {
        return "Customer [id=" + id + ", name=" + name + ", email=" + email
            + ", regDate=" + regDate + " ]";
    }
}
```

```
}
```

Après avoir créé une classe de données annotée, utilisez-la pour créer la `TableSchema`, comme indiqué dans l'extrait de code suivant.

```
static final TableSchema<Customer> customerTableSchema =  
    TableSchema.fromBean(Customer.class);
```

A `TableSchema` est conçu pour être statique et immuable. Vous pouvez généralement l'instancier au moment du chargement de la classe.

La méthode statique `TableSchema.fromBean()` factory introspecte le bean pour générer le mappage des attributs de classe de données (propriétés) vers et depuis les attributs DynamoDB.

Pour un exemple d'utilisation d'un modèle de données composé de plusieurs classes de données, consultez la `Person` classe dans la [???](#) section.

Utilisez un constructeur

Vous pouvez éviter le coût de l'introspection des haricots si vous définissez le schéma de table dans le code. Si vous codez le schéma, votre classe n'a pas besoin de suivre les normes de `JavaBean` dénomination ni d'être annotée. L'exemple suivant utilise un générateur et est équivalent à l'exemple de `Customer` classe qui utilise des annotations.

```
static final TableSchema<Customer> customerTableSchema =  
    TableSchema.builder(Customer.class)  
        .newItemSupplier(Customer::new)  
        .addAttribute(String.class, a -> a.name("id")  
            .getter(Customer::getId)  
            .setter(Customer::setId)  
            .tags(StaticAttributeTags.primaryPartitionKey()))  
        .addAttribute(String.class, a -> a.name("email")  
            .getter(Customer::getEmail)  
            .setter(Customer::setEmail)  
            .tags(StaticAttributeTags.primarySortKey()))  
        .addAttribute(String.class, a -> a.name("name")  
            .getter(Customer::getCustName)  
            .setter(Customer::setCustName))  
        .addAttribute(Instant.class, a -> a.name("registrationDate")  
            .getter(Customer::getRegistrationDate)  
            .setter(Customer::setRegistrationDate))  
        .build();
```


Créez un client amélioré et **DynamoDbTable**

Créez un client amélioré

La [DynamoDbEnhancedClient](#) classe, ou son équivalent asynchrone [DynamoDbEnhancedAsyncClient](#), est le point d'entrée pour utiliser l'API DynamoDB Enhanced Client.

Le client amélioré a besoin d'une norme [DynamoDbClient](#) pour effectuer le travail. L'API propose deux méthodes pour créer une `DynamoDbEnhancedClient` instance. La première option, illustrée dans l'extrait suivant, crée une norme dont les paramètres `DynamoDbClient` par défaut sont extraits des paramètres de configuration.

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();
```

Si vous souhaitez configurer le client standard sous-jacent, vous pouvez le fournir à la méthode de création du client amélioré, comme indiqué dans l'extrait suivant.

```
// Configure an instance of the standard DynamoDbClient.
DynamoDbClient standardClient = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

// Use the configured standard client with the enhanced client.
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
    .dynamoDbClient(standardClient)
    .build();
```

Créer une instance **DynamoDbTable**

Considérez a [DynamoDbTable](#) comme la représentation côté client d'une table DynamoDB qui utilise la fonctionnalité de mappage fournie par `TableSchema`. La `DynamoDbTable` classe fournit des méthodes pour les opérations CRUD qui vous permettent d'interagir avec une seule table DynamoDB.

`DynamoDbTable<T>` est une classe générique qui prend un seul argument de type, qu'il s'agisse d'une classe personnalisée ou d'un `EnhancedDocument` lorsque vous travaillez avec des éléments de type document. Ce type d'argument établit la relation entre la classe que vous utilisez et l'unique table DynamoDB.

Utilisez la méthode `table()` factory du `DynamoDbEnhancedClient` pour créer une `DynamoDbTable` instance, comme indiqué dans l'extrait de code suivant.

```
static final DynamoDbTable<Customer> customerTable =
    enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
```

`DynamoDbTable` instances sont candidates aux singletons car elles sont immuables et peuvent être utilisées dans l'ensemble de votre application.

Votre code possède désormais une représentation en mémoire d'une table DynamoDB qui peut fonctionner avec des instances. `Customer` La table DynamoDB réelle peut exister ou ne pas exister. Si la table nommée `Customer` déjà, vous pouvez commencer à effectuer des opérations CRUD sur elle. Si elle n'existe pas, utilisez l'`DynamoDbTable` instance pour créer la table comme indiqué dans la section suivante.

Créer une table DynamoDB si nécessaire

Après avoir créé une `DynamoDbTable` instance, utilisez-la pour créer une seule fois une table dans DynamoDB.

Créer un exemple de code de table

L'exemple suivant crée une table DynamoDB basée sur `Customer` la classe de données.

Cet exemple crée une table DynamoDB dont le nom est identique au `Customer` nom de classe, mais le nom de la table peut être différent. Quel que soit le nom que vous donnez à la table, vous devez utiliser ce nom dans d'autres applications pour travailler avec la table. Fournissez ce nom à la `table()` méthode chaque fois que vous créez un autre `DynamoDbTable` objet afin de travailler avec la table DynamoDB sous-jacente.

Le paramètre Java `lambda.builder`, transmis à la `createTable` méthode vous permet de [personnaliser le tableau](#). Dans cet exemple, le [débit provisionné](#) est configuré. Si vous souhaitez utiliser les paramètres par défaut lorsque vous créez une table, ignorez le générateur, comme indiqué dans l'extrait de code suivant.

```
customerTable.createTable();
```

Lorsque les paramètres par défaut sont utilisés, les valeurs du débit provisionné ne sont pas définies. Au lieu de cela, le mode de facturation de la table est défini [sur demande](#).

L'exemple utilise également un [DynamoDbWaiter](#) avant de tenter d'imprimer le nom de la table reçu dans la réponse. La création d'une table prend un certain temps. Par conséquent, l'utilisation d'un serveur signifie que vous n'avez pas à écrire une logique qui interroge le service DynamoDB pour vérifier si la table existe avant de l'utiliser.

Importations

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

Code

```
public static void createCustomerTable(DynamoDbTable<Customer> customerTable,
DynamoDbClient standardClient) {
    // Create the DynamoDB table using the 'customerTable' DynamoDbTable instance.
    customerTable.createTable(builder -> builder
        .provisionedThroughput(b -> b
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
    );
    // The DynamoDbClient instance (named 'standardClient') passed to the builder for
    the DynamoDbWaiter is the same instance
    // that was passed to the builder of the DynamoDbEnhancedClient instance that we
    created previously.
    // By using the same instance, it ensures that the same Region that was configured
    on the standard DynamoDbClient
    // instance is used for other service clients that accept a DynamoDbClient during
    construction.
    try (DynamoDbWaiter waiter =
        DynamoDbWaiter.builder().client(standardClient).build()) { // DynamoDbWaiter is
        Autocloseable
            ResponseOrException<DescribeTableResponse> response = waiter
                .waitUntilTableExists(builder ->
                    builder.tableName("Customer").build())
                .matched();
            DescribeTableResponse tableDescription = response.response().orElseThrow(
                () -> new RuntimeException("Customer table was not created."));
```

```
// The actual error can be inspected in response.exception()
logger.info("Customer table was created.");
}
}
```

Note

Les noms d'attributs d'une table DynamoDB commencent par une lettre minuscule lorsque la table est générée à partir d'une classe de données. Si vous souhaitez que le nom d'attribut de la table commence par une lettre majuscule, utilisez l'[@DynamoDbAttribute\(NAME\)annotation](#) et indiquez le nom souhaité en tant que paramètre.

Réaliser des opérations

Une fois la table créée, utilisez l'`DynamoDbTable` instance pour effectuer des opérations sur la table DynamoDB.

Dans l'exemple suivant, un singleton `DynamoDbTable<Customer>` est transmis en tant que paramètre avec une instance de [classe de Customer données](#) pour ajouter un nouvel élément à la table.

```
public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
    logger.info(customer.toString());
    customerTable.putItem(customer);
}
```

Objet **Customer**

```
Customer customer = new Customer();
customer.setId("1");
customer.setCustName("Customer Name");
customer.setEmail("customer@example.com");
customer.setRegistrationDate(Instant.parse("2023-07-03T10:15:30.00Z"));
```

Avant d'envoyer l'`customer` objet au service DynamoDB, enregistrez le résultat de la méthode de l'objet `toString()` pour le comparer à ce que le client amélioré envoie.

```
Customer [id=1, name=Customer Name, email=customer@example.com,  
regDate=2023-07-03T10:15:30Z]
```

La journalisation au niveau du fil indique la charge utile de la demande générée. Le client amélioré a généré la représentation de bas niveau à partir de la classe de données. L'`regDate` attribut, qui est un `Instant` type en Java, est représenté sous la forme d'une chaîne DynamoDB.

```
{  
  "TableName": "Customer",  
  "Item": {  
    "registrationDate": {  
      "S": "2023-07-03T10:15:30Z"  
    },  
    "id": {  
      "S": "1"  
    },  
    "custName": {  
      "S": "Customer Name"  
    },  
    "email": {  
      "S": "customer@example.com"  
    }  
  }  
}
```

Travailler avec une table existante

La section précédente a montré comment créer une table DynamoDB à partir d'une classe de données Java. Si vous possédez déjà une table et que vous souhaitez utiliser les fonctionnalités du client amélioré, vous pouvez créer une classe de données Java qui fonctionnera avec la table. Vous devez examiner la table DynamoDB et ajouter les annotations nécessaires à la classe de données.

Avant de travailler avec une table existante, appelez la `DynamoDbEnhanced.table()` méthode. Cela a été fait dans l'exemple précédent avec l'instruction suivante.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",  
TableSchema.fromBean(Customer.class));
```

Une fois l'`DynamoDbTable` instance renvoyée, vous pouvez commencer à travailler immédiatement avec la table sous-jacente. Il n'est pas nécessaire de recréer la table en appelant la `DynamoDbTable.createTable()` méthode.

L'exemple suivant illustre cela en récupérant immédiatement une `Customer` instance de la table DynamoDB.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",
    TableSchema.fromBean(Customer.class));
// The Customer table exists already and has an item with a primary key value of "1"
// and a sort key value of "customer@example.com".
customerTable.getItem(
    Key.builder()
        .partitionValue("1")
        .sortValue("customer@example.com").build());
```

Important

Le nom de table utilisé dans la `table()` méthode doit correspondre au nom de table DynamoDB existant.

Découvrez les principes de base de l'API client améliorée DynamoDB

Cette rubrique décrit les fonctionnalités de base de l'API DynamoDB Enhanced Client et les compare à l'API client [DynamoDB standard](#).

Si vous découvrez l'API DynamoDB Enhanced Client, nous vous recommandons de suivre [le didacticiel d'introduction pour vous familiariser avec les](#) classes fondamentales.

Éléments DynamoDB en Java

Les tables DynamoDB stockent des éléments. Selon votre cas d'utilisation, les éléments du côté Java peuvent prendre la forme de données structurées statiquement ou de structures créées dynamiquement.

Si votre cas d'utilisation nécessite des éléments dotés d'un ensemble cohérent d'attributs, utilisez des [classes annotées](#) ou utilisez un [générateur](#) pour générer le type statique `TableSchema` approprié.

Sinon, si vous devez stocker des éléments composés de différentes structures, créez un `DocumentTableSchema`. `DocumentTableSchema` fait partie de l'[API Enhanced Document](#) et ne nécessite qu'une clé primaire de type statique et fonctionne avec des `EnhancedDocument` instances pour contenir les éléments de données. L'API Enhanced Document est abordée dans une autre [rubrique](#).

Types d'attributs pour les classes de modèles de données

Bien que DynamoDB prenne [en charge un petit nombre de types d'attributs](#) par rapport au système de types riches de Java, l'API DynamoDB Enhanced Client fournit des mécanismes permettant de convertir les membres d'une classe Java vers et depuis les types d'attributs DynamoDB.

Les types d'attributs (propriétés) de vos classes de données Java doivent être des types d'objets et non des primitives. Par exemple, utilisez toujours des types de données Long et des Integer objets, long et non des int primitives.

[Par défaut, l'API DynamoDB Enhanced Client prend en charge les convertisseurs d'attributs pour un grand nombre de types, tels que Integer, BigDecimal, String et Instant.](#) La liste apparaît dans les [classes d'implémentation connues de l'AttributeConverter interface](#). La liste comprend de nombreux types et collections tels que des cartes, des listes et des ensembles.

Pour stocker les données d'un type d'attribut qui n'est pas pris en charge par défaut ou qui n'est pas conforme à la JavaBean convention, vous pouvez écrire une AttributeConverter implémentation personnalisée pour effectuer la conversion. Consultez la section sur la conversion d'attributs pour un [exemple](#).

Pour stocker les données d'un type d'attribut dont la classe est conforme à la spécification Java Beans (ou d'une [classe de données immuable](#)), vous pouvez adopter deux approches.

- Si vous avez accès au fichier source, vous pouvez annoter la classe avec @DynamoDbBean (ou @DynamoDbImmutable). La section qui traite des attributs imbriqués présente des [exemples](#) d'utilisation de classes annotées.
- Si vous n'avez pas accès au fichier source de la classe de JavaBean données pour l'attribut (ou si vous ne souhaitez pas annoter le fichier source d'une classe à laquelle vous avez accès), vous pouvez utiliser l'approche du générateur. Cela crée un schéma de table sans définir les clés. Vous pouvez ensuite imbriquer ce schéma de table dans un autre schéma de table pour effectuer le mappage. La section des attributs imbriqués contient un [exemple](#) illustrant l'utilisation de schémas imbriqués.

Valeurs nulles

Lorsque vous utilisez l'putItemAPI, le client amélioré n'inclut pas les attributs à valeur nulle d'un objet de données mappé dans la demande adressée à DynamoDB.

Pour les `updateItem` demandes, les attributs à valeur nulle sont supprimés de l'élément de la base de données. Si vous avez l'intention de mettre à jour certaines valeurs d'attribut et de conserver les autres inchangées, copiez les valeurs des autres attributs qui ne doivent pas être modifiés ou utilisez la méthode `ignoreNull()` dans le générateur de mise à jour.

L'exemple suivant illustre `ignoreNulls()` la the `updateItem()` méthode.

```
public void updateItemNullsExample(){
    Customer customer = new Customer();
    customer.setCustName("CustName");
    customer.setEmail("email");
    customer.setId("1");
    customer.setRegistrationDate(Instant.now());

    // Put item with values for all attributes.
    customerDynamoDbTable.putItem(customer);

    // Create a Customer instance with the same id value, but a different name
value.
    // Do not set the 'registrationDate' attribute.
    Customer custForUpdate = new Customer();
    custForUpdate.setCustName("NewName");
    custForUpdate.setEmail("email");
    custForUpdate.setId("1");

    // Update item without setting the registrationDate attribute.
    customerDynamoDbTable.updateItem(b -> b
        .item(custForUpdate)
        .ignoreNulls(Boolean.TRUE));

    Customer updatedWithNullsIgnored = customerDynamoDbTable.getItem(customer);
    // registrationDate value is unchanged.
    logger.info(updatedWithNullsIgnored.toString());

    customerDynamoDbTable.updateItem(custForUpdate);
    Customer updatedWithNulls = customerDynamoDbTable.getItem(customer);
    // registrationDate value is null because ignoreNulls() was not used.
    logger.info(updatedWithNulls.toString());
}
}

// Logged lines.
```



```
Customer [id=1, custName=NewName, email=email,
  registrationDate=2023-04-05T16:32:32.056Z]
Customer [id=1, custName=NewName, email=email, registrationDate=null]
```

Méthodes de base du client DynamoDB Enhanced

Les méthodes de base du client amélioré correspondent aux opérations de service DynamoDB dont elles portent le nom. Les exemples suivants montrent la variante la plus simple de chaque méthode. Vous pouvez personnaliser chaque méthode en transmettant un objet de demande amélioré. Les objets de requête améliorés offrent la plupart des fonctionnalités disponibles dans le client DynamoDB standard. Ils sont entièrement documentés dans le Guide de référence des AWS SDK for Java 2.x API.

L'exemple utilise ce qui [the section called "classe Customer"](#) est indiqué précédemment.

```
// CreateTable
customerTable.createTable();

// GetItem
Customer customer =
  customerTable.getItem(Key.builder().partitionValue("a123").build());

// UpdateItem
Customer updatedCustomer = customerTable.updateItem(customer);

// PutItem
customerTable.putItem(customer);

// DeleteItem
Customer deletedCustomer =
  customerTable.deleteItem(Key.builder().partitionValue("a123").sortValue(456).build());

// Query
PageIterable<Customer> customers = customerTable.query(keyEqualTo(k ->
  k.partitionValue("a123")));

// Scan
PageIterable<Customer> customers = customerTable.scan();

// BatchGetItem
BatchGetResultPageIterable batchResults =
  enhancedClient.batchGetItem(r -> r.addReadBatch(ReadBatch.builder(Customer.class)
```

```

        .mappedTableResource(customerTable)
        .addGetItem(key1)
        .addGetItem(key2)
        .addGetItem(key3)
        .build());

// BatchWriteItem
batchResults = enhancedClient.batchWriteItem(r ->
    r.addWriteBatch(WriteBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addPutItem(customer)
        .addDeleteItem(key1)
        .addDeleteItem(key1)
        .build()));

// TransactGetItems
transactResults = enhancedClient.transactGetItems(r -> r.addGetItem(customerTable,
    key1)
    .addGetItem(customerTable,
    key2));

// TransactWriteItems
enhancedClient.transactWriteItems(r -> r.addConditionCheck(customerTable,
    i -> i.key(orderKey)
    .conditionExpression(conditionExpression))
    .addUpdateItem(customerTable, customer)
    .addDeleteItem(customerTable, key));

```

Comparez le client DynamoDB amélioré au client DynamoDB standard

Les API clientes DynamoDB (standard [et](#) améliorées) vous [permettent](#) de travailler avec des tables DynamoDB pour effectuer des opérations CRUD (création, lecture, mise à jour et suppression) au niveau des données. La différence entre les API clientes réside dans la manière dont cela est réalisé. Avec le client standard, vous travaillez directement avec des attributs de données de bas niveau. L'API client améliorée utilise des classes Java familières et correspond à l'API de bas niveau utilisée en arrière-plan.

Alors que les deux API client prennent en charge les opérations au niveau des données, le client DynamoDB standard prend également en charge les opérations au niveau des ressources. Les opérations au niveau des ressources gèrent la base de données, telles que la création de sauvegardes, la liste des tables et la mise à jour des tables. L'API client améliorée prend en charge

un certain nombre d'opérations au niveau des ressources, telles que la création, la description et la suppression de tables.

Pour illustrer les différentes approches utilisées par les deux API clientes, les exemples de code suivants montrent la création de la même ProductCatalog table à l'aide du client standard et du client amélioré.

Comparaison : création d'une table à l'aide du client DynamoDB standard

```

DependencyFactory.dynamoDbClient().createTable(builder -> builder
    .tableName(TABLE_NAME)
    .attributeDefinitions(
        b -> b.attributeName("id").attributeType(ScalarAttributeType.N),
        b -> b.attributeName("title").attributeType(ScalarAttributeType.S),
        b -> b.attributeName("isbn").attributeType(ScalarAttributeType.S)
    )
    .keySchema(
        builder1 -> builder1.attributeName("id").keyType(KeyType.HASH),
        builder2 -> builder2.attributeName("title").keyType(KeyType.RANGE)
    )
    .globalSecondaryIndexes(builder3 -> builder3
        .indexName("products_by_isbn")
        .keySchema(builder2 -> builder2
            .attributeName("isbn").keyType(KeyType.HASH))
        .projection(builder2 -> builder2
            .projectionType(ProjectionType.INCLUDE)
            .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(builder4 -> builder4
            .writeCapacityUnits(5L).readCapacityUnits(5L))
    )
    .provisionedThroughput(builder1 -> builder1
        .readCapacityUnits(5L).writeCapacityUnits(5L))
);

```

Comparaison : création d'une table à l'aide du client DynamoDB Enhanced

```

DynamoDbEnhancedClient enhancedClient = DependencyFactory.enhancedClient();
productCatalog = enhancedClient.table(TABLE_NAME,
    TableSchema.fromImmutableClass(ProductCatalog.class));
productCatalog.createTable(b -> b
    .provisionedThroughput(b1 -> b1.readCapacityUnits(5L).writeCapacityUnits(5L))
    .globalSecondaryIndices(b2 -> b2.indexName("products_by_isbn")
        .projection(b4 -> b4

```

```

                .projectionType(ProjectionType.INCLUDE)
                .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(b3 ->
b3.writeCapacityUnits(5L).readCapacityUnits(5L))
        )
);

```

Le client amélioré utilise la classe de données annotée suivante. Le client DynamoDB Enhanced fait correspondre les types de données Java aux types de données DynamoDB pour un code moins détaillé et plus facile à suivre. `ProductCatalog` est un exemple d'utilisation d'une classe immuable avec le client DynamoDB Enhanced. L'utilisation de classes immuables pour les classes de données mappées est [abordée plus loin](#) dans cette rubrique.

classe **ProductCatalog**

```

package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbIgnore;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.math.BigDecimal;
import java.util.Objects;
import java.util.Set;

@DynamoDbImmutable(builder = ProductCatalog.Builder.class)
public class ProductCatalog implements Comparable<ProductCatalog> {
    private Integer id;
    private String title;
    private String isbn;
    private Set<String> authors;
    private BigDecimal price;

    private ProductCatalog(Builder builder){
        this.authors = builder.authors;
        this.id = builder.id;
        this.isbn = builder.isbn;
        this.price = builder.price;
    }
}

```

```
        this.title = builder.title;
    }

    public static Builder builder(){ return new Builder(); }

    @DynamoDbPartitionKey
    public Integer id() { return id; }

    @DynamoDbSortKey
    public String title() { return title; }

    @DynamoDbSecondaryPartitionKey(indexNames = "products_by_isbn")
    public String isbn() { return isbn; }
    public Set<String> authors() { return authors; }
    public BigDecimal price() { return price; }

    public static final class Builder {
        private Integer id;
        private String title;
        private String isbn;
        private Set<String> authors;
        private BigDecimal price;
        private Builder(){}

        public Builder id(Integer id) { this.id = id; return this; }
        public Builder title(String title) { this.title = title; return this; }
        public Builder isbn(String ISBN) { this.isbn = ISBN; return this; }
        public Builder authors(Set<String> authors) { this.authors = authors; return
this; }
        public Builder price(BigDecimal price) { this.price = price; return this; }
        public ProductCatalog build() { return new ProductCatalog(this); }
    }

    @Override
    public String toString() {
        final StringBuffer sb = new StringBuffer("ProductCatalog{");
        sb.append("id=").append(id);
        sb.append(", title=").append(title).append('\ ');
        sb.append(", isbn=").append(isbn).append('\ ');
        sb.append(", authors=").append(authors);
        sb.append(", price=").append(price);
        sb.append('}');
        return sb.toString();
    }
}
```

```

    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        ProductCatalog that = (ProductCatalog) o;
        return id.equals(that.id) && title.equals(that.title) && Objects.equals(isbn,
that.isbn) && Objects.equals(authors, that.authors) && Objects.equals(price,
that.price);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, title, isbn, authors, price);
    }

    @Override
    @DynamoDbIgnore
    public int compareTo(ProductCatalog other) {
        if (this.id.compareTo(other.id) != 0){
            return this.id.compareTo(other.id);
        } else {
            return this.title.compareTo(other.title);
        }
    }
}
}

```

Les deux exemples de code suivants illustrant une écriture par lots illustrent le caractère verbeux et le manque de sécurité de type liés à l'utilisation du client standard par opposition au client amélioré.

Comparaison : écriture par lots à l'aide du client DynamoDB standard

```

public static void batchWriteStandard(DynamoDbClient dynamoDbClient, String
tableName) {

    Map<String, AttributeValue> catalogItem = Map.of(
        "authors", AttributeValue.builder().ss("a", "b").build(),
        "id", AttributeValue.builder().n("1").build(),
        "isbn", AttributeValue.builder().s("1-565-85698").build(),
        "title", AttributeValue.builder().s("Title 1").build(),
        "price", AttributeValue.builder().n("52.13").build());
}

```

```

Map<String, AttributeValue> catalogItem2 = Map.of(
    "authors", AttributeValue.builder().ss("a", "b", "c").build(),
    "id", AttributeValue.builder().n("2").build(),
    "isbn", AttributeValue.builder().s("1-208-98073").build(),
    "title", AttributeValue.builder().s("Title 2").build(),
    "price", AttributeValue.builder().n("21.99").build());

Map<String, AttributeValue> catalogItem3 = Map.of(
    "authors", AttributeValue.builder().ss("g", "k", "c").build(),
    "id", AttributeValue.builder().n("3").build(),
    "isbn", AttributeValue.builder().s("7-236-98618").build(),
    "title", AttributeValue.builder().s("Title 3").build(),
    "price", AttributeValue.builder().n("42.00").build());

Set<WriteRequest> writeRequests = Set.of(
    WriteRequest.builder().putRequest(b -> b.item(catalogItem)).build(),
    WriteRequest.builder().putRequest(b -> b.item(catalogItem2)).build(),
    WriteRequest.builder().putRequest(b -> b.item(catalogItem3)).build());

Map<String, Set<WriteRequest>> productCatalogItems = Map.of(
    "ProductCatalog", writeRequests);

BatchWriteItemResponse response = dynamoDbClient.batchWriteItem(b ->
b.requestItems(productCatalogItems));

logger.info("Unprocessed items: " + response.unprocessedItems().size());
}

```

Comparaison : écriture par lots à l'aide du client DynamoDB Enhanced

```

public static void batchWriteEnhanced(DynamoDbTable<ProductCatalog> productCatalog)
{
    ProductCatalog prod = ProductCatalog.builder()
        .id(1)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(52.13))
        .title("Title 1")
        .build();
    ProductCatalog prod2 = ProductCatalog.builder()
        .id(2)
        .isbn("1-208-98073")
        .authors(new HashSet<>(Arrays.asList("a", "b", "c")))

```

```

        .price(BigDecimal.valueOf(21.99))
        .title("Title 2")
        .build();
ProductCatalog prod3 = ProductCatalog.builder()
    .id(3)
    .isbn("7-236-98618")
    .authors(new HashSet<>(Arrays.asList("g", "k", "c")))
    .price(BigDecimal.valueOf(42.00))
    .title("Title 3")
    .build();

BatchWriteResult batchWriteResult = DependencyFactory.enhancedClient()
    .batchWriteItem(b -> b.writeBatches(
        WriteBatch.builder(ProductCatalog.class)
            .mappedTableResource(productCatalog)
            .addPutItem(prod).addPutItem(prod2).addPutItem(prod3)
            .build()
    ));
logger.info("Unprocessed items: " +
batchWriteResult.unprocessedPutItemsForTable(productCatalog).size());
}

```

Travaillez avec des classes de données immuables

La fonctionnalité de mappage de l'API DynamoDB Enhanced Client fonctionne avec des classes de données immuables. Une classe immuable ne possède que des getters et nécessite une classe de générateur que le SDK utilise pour créer des instances de la classe. Au lieu d'utiliser l'@DynamoDbBeanannotation comme indiqué dans la [classe Customer](#), les classes immuables utilisent l'@DynamoDbImmutableannotation, qui prend un paramètre indiquant la classe de générateur à utiliser.

La classe suivante est une version immuable de Customer

```

package org.example.tests.model.immutable;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

```



```
import java.time.Instant;

@DynamoDbImmutable(builder = CustomerImmutable.Builder.class)
public class CustomerImmutable {
    private final String id;
    private final String name;
    private final String email;
    private final Instant regDate;

    private CustomerImmutable(Builder b) {
        this.id = b.id;
        this.email = b.email;
        this.name = b.name;
        this.regDate = b.regDate;
    }

    // This method will be automatically discovered and used by the TableSchema.
    public static Builder builder() { return new Builder(); }

    @DynamoDbPartitionKey
    public String id() { return this.id; }

    @DynamoDbSortKey
    public String email() { return this.email; }

    @DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name")
    public String name() { return this.name; }

    @DynamoDbSecondarySortKey(indexNames = {"customers_by_date", "customers_by_name"})
    public Instant regDate() { return this.regDate; }

    public static final class Builder {
        private String id;
        private String email;
        private String name;
        private Instant regDate;

        // The private Builder constructor is visible to the enclosing
        CustomerImmutable class.
        private Builder() {}

        public Builder id(String id) { this.id = id; return this; }
        public Builder email(String email) { this.email = email; return this; }
```

```
public Builder name(String name) { this.name = name; return this; }
public Builder regDate(Instant regDate) { this.regDate = regDate; return
this; }

// This method will be automatically discovered and used by the TableSchema.
public CustomerImmutable build() { return new CustomerImmutable(this); }
}
}
```

Vous devez satisfaire aux exigences suivantes lorsque vous annotez une classe de données avec `@DynamoDbImmutable`.

1. Chaque méthode qui n'est pas une substitution `Object.class` et qui n'a pas été annotée `@DynamoDbIgnore` doit être un getter pour un attribut de la table DynamoDB.
2. Chaque getter doit avoir un setter correspondant distinguant majuscules et minuscules dans la classe de générateur.
3. Seule l'une des conditions de construction suivantes doit être respectée.
 - La classe de générateur doit avoir un constructeur public par défaut.
 - La classe de données doit avoir une méthode statique publique nommée `builder()` qui ne prend aucun paramètre et renvoie une instance de la classe de générateur. Cette option est affichée dans la `Customer` classe immuable.
4. La classe builder doit avoir une méthode publique nommée `build()` qui ne prend aucun paramètre et renvoie une instance de la classe immuable.

Pour créer un `TableSchema` pour votre classe immuable, utilisez la `fromImmutableClass()` méthode on `TableSchema` comme indiqué dans l'extrait suivant.

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =
    TableSchema.fromImmutableClass(CustomerImmutable.class);
```

Tout comme vous pouvez créer une table DynamoDB à partir d'une classe mutable, vous pouvez en créer une à partir d'une classe immuable avec un appel unique à `of`, comme indiqué dans l'exemple `DynamoDbTable` d'`createTable()` extrait de code suivant.

```
static void createTableFromImmutable(DynamoDbEnhancedClient enhancedClient, String
tableName, DynamoDbWaiter waiter){
    // First, create an in-memory representation of the table using the 'table()'
method of the DynamoDb Enhanced Client.
```

```

// 'table()' accepts a name for the table and a TableSchema instance that you
created previously.
DynamoDbTable<CustomerImmutable> customerDynamoDbTable = enhancedClient
    .table(tableName, TableSchema.fromImmutableClass(CustomerImmutable.class));

// Second, call the 'createTable()' method on the DynamoDbTable instance.
customerDynamoDbTable.createTable();
waiter.waitForTableExists(b -> b.tableName(tableName));
}

```

Utilisez des bibliothèques tierces, telles que Lombok

Les bibliothèques tierces, telles que [Project Lombok](#), aident à générer du code standard associé à des objets immuables. L'API DynamoDB Enhanced Client fonctionne avec ces bibliothèques tant que les classes de données respectent les conventions détaillées dans cette section.

L'exemple suivant montre la CustomerImmutable classe immuable avec les annotations Lombok. Notez comment la onMethod fonctionnalité de Lombok copie les annotations DynamoDB basées sur les attributs, telles que, sur le code généré. @DynamoDbPartitionKey

```

@Value
@Builder
@dynamoDbImmutable(builder = Customer.CustomerBuilder.class)
public class Customer {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;

    @Getter(onMethod_=@DynamoDbSortKey)
    private String email;

    @Getter(onMethod_=@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name"))
    private String name;

    @Getter(onMethod_=@DynamoDbSecondarySortKey(indexNames = {"customers_by_date",
"customers_by_name"}))
    private Instant createdAt;
}

```

Expressions et conditions d'utilisation

[Les expressions de l'API DynamoDB Enhanced Client sont des représentations Java des expressions DynamoDB.](#)

L'API client améliorée DynamoDB utilise trois types d'expressions :

Expression

La Expression classe est utilisée lorsque vous définissez des conditions et des filtres.

QueryConditional

Ce type d'expression représente [les conditions clés](#) pour les opérations de requête.

UpdateExpression

Cette classe vous aide à écrire des expressions de [mise à jour DynamoDB](#) et est actuellement utilisée dans le framework d'extension lorsque vous mettez à jour un élément.

Anatomie de l'expression

Une expression est composée des éléments suivants :

- Une expression sous forme de chaîne (obligatoire). La chaîne contient une expression logique DynamoDB avec des noms d'espaces réservés pour les noms d'attributs et les valeurs d'attribut.
- Une carte des valeurs d'expression (généralement obligatoire).
- Carte des noms d'expressions (facultatif).

Utilisez un générateur pour générer un Expression objet qui prend la forme générale suivante.

```
Expression expression = Expression.builder()
    .expression(<String>)
    .expressionNames(<Map>)
    .expressionValues(<Map>)
    .build()
```

Expressions nécessitent généralement une carte des valeurs d'expression. La carte fournit les valeurs des espaces réservés dans l'expression sous forme de chaîne. La clé de carte se compose du nom de l'espace réservé précédé de deux points (:) et la valeur de la carte est une instance de [AttributeValue](#). La [AttributeValues](#) classe dispose de méthodes pratiques pour générer une AttributeValue instance à partir d'un littéral. Vous pouvez également utiliser le AttributeValue.Builder pour générer une AttributeValue instance.

L'extrait suivant montre une carte avec deux entrées après la ligne de commentaire 2. La chaîne transmise à la expression() méthode, affichée après la ligne de commentaire 1, contient les

espaces réservés que DynamoDB résout avant d'effectuer l'opération. Cet extrait ne contient pas de carte des noms d'expressions, car le prix est un nom d'attribut autorisé.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
            // provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
            // supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();
}
```

Si un nom d'attribut dans la table DynamoDB est un mot réservé, commence par un chiffre ou contient un espace, une carte des noms d'expressions est requise pour le. Expression

Par exemple, si le nom de l'attribut ne figurait *1price* pas *price* dans l'exemple de code précédent, celui-ci devra être modifié comme indiqué dans l'exemple suivant.

```
ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .filterExpression(Expression.builder()
        .expression("#price >= :min_value AND #price <= :max_value")
        .expressionNames( Map.of("#price", "1price") )
        .expressionValues(
            Map.of(":min_value", numberValue(8.00),
                ":max_value", numberValue(400_000.00)))
        .build())
    .build();
```

Un espace réservé pour le nom d'une expression commence par le signe dièse (#). Une entrée pour la carte des noms d'expressions utilise l'espace réservé comme clé et le nom de l'attribut comme valeur. La carte est ajoutée au générateur d'expressions avec la `expressionNames()` méthode. DynamoDB résout le nom de l'attribut avant d'effectuer l'opération.

Les valeurs d'expression ne sont pas obligatoires si une fonction est utilisée dans l'expression sous forme de chaîne. Voici un exemple de fonction d'expression `attribute_exists(<attribute_name>)`.

L'exemple suivant crée un fichier `Expression` qui utilise une fonction [DynamoDB](#). Dans cet exemple, la chaîne d'expression n'utilise aucun espace réservé. Cette expression peut être utilisée lors d'une `putItem` opération visant à vérifier si un élément existe déjà dans la base de données avec une valeur d'`movieattribut` égale à celle de l'`movieattribut` de l'objet de données.

```
Expression exp = Expression.builder().expression("attribute_not_exists\n(movie)").build();
```

Le guide du développeur DynamoDB contient des informations complètes sur les expressions de [bas niveau utilisées](#) avec DynamoDB.

Expressions de conditions et conditions

Lorsque vous utilisez les `deleteItem()` méthodes `putItem()`, et `updateItem()`, ainsi que lorsque vous utilisez des opérations de transaction et de traitement par lots, vous utilisez [Expression](#) des objets pour spécifier les conditions que DynamoDB doit respecter pour poursuivre l'opération. Ces expressions sont appelées expressions de condition. Pour un exemple, consultez l'expression de condition utilisée dans la `addDeleteItem()` méthode (après la ligne de commentaire 1) de l'[exemple de transaction](#) présenté dans ce guide.

Lorsque vous utilisez les `query()` méthodes, une condition est exprimée sous la forme d'un [QueryConditional](#). La `QueryConditional` classe dispose de plusieurs méthodes pratiques statiques qui vous aident à écrire les critères qui déterminent les éléments à lire dans DynamoDB.

Pour des exemples `QueryConditionals`, consultez le premier exemple de code de la [the section called "Queryexemples de méthodes"](#) section de ce guide.

Expressions de filtrage

Les expressions de filtre sont utilisées dans les opérations de numérisation et de requête pour filtrer les éléments renvoyés.

Une expression de filtre est appliquée une fois que toutes les données ont été lues dans la base de données. Le coût de lecture est donc le même que s'il n'y avait pas de filtre. [Le guide du développeur Amazon DynamoDB contient plus d'informations sur l'utilisation d'expressions de filtre pour les opérations de requête et de numérisation.](#)

L'exemple suivant montre une expression de filtre ajoutée à une demande d'analyse. Les critères limitent les articles retournés aux articles dont le prix se situe entre 8,00 et 80,00€ inclus.

```
Map<String, AttributeValue> expressionValues = Map.of(
    ":min_value", numberValue(8.00),
    ":max_value", numberValue(80.00));

ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .consistentRead(true)
    // 1. the 'attributesToProject()' method allows you to specify which
    values you want returned.
    .attributesToProject("id", "title", "authors", "price")
    // 2. Filter expression limits the items returned that match the
    provided criteria.
    .filterExpression(Expression.builder()
        .expression("price >= :min_value AND price <= :max_value")
        .expressionValues(expressionValues)
        .build())
    .build();
```

Expressions de mise à jour

La méthode du client `updateItem()` DynamoDB Enhanced fournit une méthode standard pour mettre à jour des éléments dans DynamoDB. Toutefois, lorsque vous avez besoin de fonctionnalités supplémentaires, [UpdateExpressions](#) fournit une représentation sécurisée de la syntaxe de l'expression de mise à jour [DynamoDB](#). Par exemple, vous pouvez l'utiliser `UpdateExpressions` pour augmenter les valeurs sans lire au préalable les éléments de DynamoDB, ou pour ajouter des membres individuels à une liste. Les expressions de mise à jour sont actuellement disponibles dans les extensions personnalisées de la `updateItem()` méthode.

Pour un exemple utilisant des expressions de mise à jour, consultez l'[exemple d'extension personnalisée](#) présenté dans ce guide.

De plus amples informations sur les expressions de mise à jour sont disponibles dans le [manuel du développeur Amazon DynamoDB](#).

Travailler avec des résultats paginés : scans et requêtes

Les *scan batch* méthodes *query* et de l'API DynamoDB Enhanced Client renvoient des réponses contenant une ou plusieurs pages. Une page contient un ou plusieurs éléments. Votre code peut traiter la réponse page par page ou traiter des éléments individuels.

Une réponse paginée renvoyée par le `DynamoDbEnhancedClient` client synchrone renvoie un [PageIterable](#) objet, tandis qu'une réponse renvoyée par le client asynchrone `DynamoDbEnhancedAsyncClient` renvoie un objet. [PagePublisher](#)

Cette section traite du traitement des résultats paginés et fournit des exemples d'utilisation des API de numérisation et de requête.

Analyser une table

La [scan](#) méthode du SDK correspond à l'opération [DynamoDB](#) du même nom. L'API `DynamoDB Enhanced Client` propose les mêmes options, mais elle utilise un modèle d'objet familier et gère la pagination à votre place.

Tout d'abord, nous explorons l'`PageIterable` interface en examinant la `scan` méthode de la classe de mappage synchrone, [DynamoDbTable](#).

Utiliser l'API synchrone

L'exemple suivant montre la `scan` méthode qui utilise une [expression](#) pour filtrer les éléments renvoyés. [ProductCatalog](#) s'agit de l'objet modèle présenté précédemment.

L'expression de filtrage affichée après la ligne de commentaire 1 limite les `ProductCatalog` articles renvoyés à ceux dont le prix est compris entre 8,00 et 80,00€ inclusivement.

Cet exemple exclut également les `isbn` valeurs en utilisant la `attributesToProject` méthode indiquée après la ligne de commentaire 2.

Sur la ligne de commentaire 3, l'`PageIterable` objet `pagedResult`, est renvoyé par la `scan` méthode. La `stream` méthode de `PageIterable` renvoi d'un [java.util.Stream](#) objet, que vous pouvez utiliser pour traiter les pages. Dans cet exemple, le nombre de pages est compté et enregistré.

À partir de la ligne de commentaire 4, l'exemple montre deux variantes d'accès aux `ProductCatalog` éléments. La version située après la ligne de commentaire 2a parcourt chaque page et trie et enregistre les éléments de chaque page. La version située après la ligne de commentaire 2b ignore l'itération de la page et accède directement aux éléments.

L'`PageIterable` interface offre plusieurs manières de traiter les résultats grâce à ses deux interfaces parentes : [java.lang.Iterable](#) et [SdkIterable](#). `Iterable` apporte les `forEach` `splitIterator` méthodes, et `SdkIterable` apporte la `stream` méthode. `iterator`


```

public static void scanSync(DynamoDbTable<ProductCatalog> productCatalog) {

    Map<String, AttributeValue> expressionValues = Map.of(
        ":min_value", numberValue(8.00),
        ":max_value", numberValue(80.00));

    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        // 1. the 'attributesToProject()' method allows you to specify which
values you want returned.
        .attributesToProject("id", "title", "authors", "price")
        // 2. Filter expression limits the items returned that match the
provided criteria.
        .filterExpression(Expression.builder()
            .expression("price >= :min_value AND price <= :max_value")
            .expressionValues(expressionValues)
            .build())
        .build();

    // 3. A PageIterable object is returned by the scan method.
    PageIterable<ProductCatalog> pagedResults = productCatalog.scan(request);
    logger.info("page count: {}", pagedResults.stream().count());

    // 4. Log the returned ProductCatalog items using two variations.
    // 4a. This version sorts and logs the items of each page.
    pagedResults.stream().forEach(p -> p.items().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(
            item -> logger.info(item.toString())
        ));
    // 4b. This version sorts and logs all items for all pages.
    pagedResults.items().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(
            item -> logger.info(item.toString())
        );
}

```

Utiliser l'API asynchrone

La scan méthode asynchrone renvoie les résultats sous forme d'`PagePublisher` objet. L'`PagePublisher` interface comporte deux `subscribe` méthodes que vous pouvez utiliser pour traiter les pages de réponse. L'une des `subscribe` méthodes provient de l'interface

`org.reactivestreams.Publisher` parent. Pour traiter les pages à l'aide de cette première option, transmettez une [Subscriber](#) instance à la `subscribe` méthode. Le premier exemple qui suit montre l'utilisation de la `subscribe` méthode.

La deuxième `subscribe` méthode provient de l'[SdkPublisher](#) interface. Cette version de `subscribe` accepte un [Consumer](#) plutôt qu'un `Subscriber`. Cette variante de `subscribe` méthode est illustrée dans le deuxième exemple qui suit.

L'exemple suivant montre la version asynchrone de la `scan` méthode qui utilise la même expression de filtre que dans l'exemple précédent.

Après la ligne de commentaire 3, `DynamoDbAsyncTable.scan` renvoie un `PagePublisher` objet. Sur la ligne suivante, le code crée une instance de l'`org.reactivestreams.SubscriberinterfaceProductCatalogSubscriber`, qui s'abonne à la ligne de commentaire 4 `PagePublisher` après.

L'`Subscriber` objet collecte les `ProductCatalog` éléments de chaque page de la `onNext` méthode après la ligne de commentaire 8 dans l'exemple `ProductCatalogSubscriber` de classe. Les éléments sont stockés dans la `List` variable privée et sont accessibles dans le code d'appel avec la `ProductCatalogSubscriber.getSubscribedItems()` méthode. Ceci est appelé après la ligne de commentaire 5.

Une fois la liste récupérée, le code trie tous les `ProductCatalog` articles par prix et enregistre chaque article.

La `ProductCatalogSubscriber` classe [CountDownLatch](#) in bloque le fil d'appel jusqu'à ce que tous les éléments aient été ajoutés à la liste avant de continuer après la ligne de commentaire 5.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
        )
    }
}
```

```

        .build())
    .build();

    // 3. A PagePublisher object is returned by the scan method.
    PagePublisher<ProductCatalog> pagePublisher = productCatalog.scan(request);
    ProductCatalogSubscriber subscriber = new ProductCatalogSubscriber();
    // 4. Subscribe the ProductCatalogSubscriber to the PagePublisher.
    pagePublisher.subscribe(subscriber);
    // 5. Retrieve all collected ProductCatalog items accumulated by the
subscriber.
    subscriber.getSubscribedItems().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString()));
    // 6. Use a Consumer to work through each page.
    pagePublisher.subscribe(page -> page
        .items().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString())))
        .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
    // 7. Use a Consumer to work through each ProductCatalog item.
    pagePublisher.items()
        .subscribe(product -> logger.info(product.toString()))
        .exceptionally(failure -> {
            logger.error("ERROR - ", failure);
            return null;
        })
        .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
    }
}

```

```

private static class ProductCatalogSubscriber implements
Subscriber<Page<ProductCatalog>> {
    private CountDownLatch latch = new CountDownLatch(1);
    private Subscription subscription;
    private List<ProductCatalog> itemsFromAllPages = new ArrayList<>();

    @Override
    public void onSubscribe(Subscription sub) {
        subscription = sub;
        subscription.request(1L);
    }
}

```

```

        try {
            latch.await(); // Called by main thread blocking it until latch is
released.
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void onNext(Page<ProductCatalog> productCatalogPage) {
        // 8. Collect all the ProductCatalog instances in the page, then ask the
publisher for one more page.
        itemsFromAllPages.addAll(productCatalogPage.items());
        subscription.request(1L);
    }

    @Override
    public void onError(Throwable throwable) {
    }

    @Override
    public void onComplete() {
        latch.countDown(); // Call by subscription thread; latch releases.
    }

    List<ProductCatalog> getSubscribedItems() {
        return this.itemsFromAllPages;
    }
}

```

L'exemple d'extrait de code suivant utilise la version de la `PagePublisher.subscribe` méthode qui accepte une ligne de commentaire `Consumer` 6 après. Le paramètre Java lambda consomme des pages, qui traitent ensuite chaque élément. Dans cet exemple, chaque page est traitée et les éléments de chaque page sont triés puis enregistrés.

```

// 6. Use a Consumer to work through each page.
pagePublisher.subscribe(page -> page
    .items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(item ->
        logger.info(item.toString()))

```

```
.join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
```

La `items` méthode qui consiste à `PagePublisher` déballer les instances du modèle afin que votre code puisse traiter les éléments directement. Cette approche est illustrée dans l'extrait suivant.

```
// 7. Use a Consumer to work through each ProductCatalog item.
pagePublisher.items()
    .subscribe(product -> logger.info(product.toString()))
    .exceptionally(failure -> {
        logger.error("ERROR - ", failure);
        return null;
    })
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
```

Interroger une table

La `query()` méthode de la `DynamoDbTable` classe trouve des éléments en fonction des valeurs des clés primaires. L'`@DynamoDbPartitionKey` annotation et l'`@DynamoDbSortKey` annotation facultative sont utilisées pour définir la clé primaire de votre classe de données.

La `query()` méthode nécessite une valeur de clé de partition qui trouve les éléments correspondant à la valeur fournie. Si votre table définit également une clé de tri, vous pouvez ajouter une valeur à cette clé à votre requête en tant que condition de comparaison supplémentaire pour affiner les résultats.

À l'exception du traitement des résultats, les versions synchrone et asynchrone de `query()` fonctionnent de la même manière. Comme pour l'`scanAPI`, l'`queryAPI` renvoie un `PageIterable` pour un appel synchrone et un `PagePublisher` pour un appel asynchrone. Nous avons discuté de l'utilisation de `PageIterable` et `PagePublisher` précédemment dans la section de numérisation.

Query exemples de méthodes

L'exemple de code de `query()` méthode qui suit utilise la `MovieActor` classe. La classe de données définit une clé primaire composite composée de l'`movie` attribut de la clé de partition et de l'`actor` attribut de la clé de tri.

La classe indique également qu'elle utilise un index secondaire global nommé `acting_award_year`. La clé primaire composite de l'index est composée de l'`actingaward` attribut de la clé de partition et de l'attribut `actingyear` de la clé de tri. Plus loin dans cette rubrique,

lorsque nous montrerons comment créer et utiliser des index, nous ferons référence à `acting_award_year` index.

classe `MovieActor`

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbAttribute;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.Objects;

@DynamoDbBean
public class MovieActor implements Comparable<MovieActor> {

    private String movieName;
    private String actorName;
    private String actingAward;
    private Integer actingYear;
    private String actingSchoolName;

    @DynamoDbPartitionKey
    @DynamoDbAttribute("movie")
    public String getMovieName() {
        return movieName;
    }

    public void setMovieName(String movieName) {
        this.movieName = movieName;
    }

    @DynamoDbSortKey
    @DynamoDbAttribute("actor")
    public String getActorName() {
        return actorName;
    }
}
```

```
public void setActorName(String actorName) {
    this.actorName = actorName;
}

@DynamoDbSecondaryPartitionKey(indexNames = "acting_award_year")
@DynamoDbAttribute("actingaward")
public String getActingAward() {
    return actingAward;
}

public void setActingAward(String actingAward) {
    this.actingAward = actingAward;
}

@DynamoDbSecondarySortKey(indexNames = {"acting_award_year", "movie_year"})
@DynamoDbAttribute("actingyear")
public Integer getActingYear() {
    return actingYear;
}

public void setActingYear(Integer actingYear) {
    this.actingYear = actingYear;
}

@DynamoDbAttribute("actingschoolname")
public String getActingSchoolName() {
    return actingSchoolName;
}

public void setActingSchoolName(String actingSchoolName) {
    this.actingSchoolName = actingSchoolName;
}

@Override
public String toString() {
    final StringBuffer sb = new StringBuffer("MovieActor{");
    sb.append("movieName=").append(movieName).append('\ ');
    sb.append(", actorName=").append(actorName).append('\ ');
    sb.append(", actingAward=").append(actingAward).append('\ ');
    sb.append(", actingYear=").append(actingYear);
    sb.append(", actingSchoolName=").append(actingSchoolName).append('\ ');
    sb.append('}');
    return sb.toString();
}
```

```

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    MovieActor that = (MovieActor) o;
    return Objects.equals(movieName, that.movieName) && Objects.equals(actorName,
that.actorName) && Objects.equals(actingAward, that.actingAward) &&
Objects.equals(actingYear, that.actingYear) && Objects.equals(actingSchoolName,
that.actingSchoolName);
}

@Override
public int hashCode() {
    return Objects.hash(movieName, actorName, actingAward, actingYear,
actingSchoolName);
}

@Override
public int compareTo(MovieActor o) {
    if (this.movieName.compareTo(o.movieName) != 0){
        return this.movieName.compareTo(o.movieName);
    } else {
        return this.actorName.compareTo(o.actorName);
    }
}
}

```

Les exemples de code qui suivent portent sur les éléments suivants.

Éléments du **MovieActor** tableau

```

MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
actingYear=2001, actingSchoolName='actingschool4'}
MovieActor{movieName='movie02', actorName='actor0', actingAward='actingaward0',
actingYear=2002, actingSchoolName='null'}

```



```

MovieActor{movieName='movie02', actorName='actor1', actingAward='actingaward1',
  actingYear=2002, actingSchoolName='actingschool1'}
MovieActor{movieName='movie02', actorName='actor2', actingAward='actingaward2',
  actingYear=2002, actingSchoolName='actingschool2'}
MovieActor{movieName='movie02', actorName='actor3', actingAward='actingaward3',
  actingYear=2002, actingSchoolName='null'}
MovieActor{movieName='movie02', actorName='actor4', actingAward='actingaward4',
  actingYear=2002, actingSchoolName='actingschool4'}
MovieActor{movieName='movie03', actorName='actor0', actingAward='actingaward0',
  actingYear=2003, actingSchoolName='null'}
MovieActor{movieName='movie03', actorName='actor1', actingAward='actingaward1',
  actingYear=2003, actingSchoolName='actingschool1'}
MovieActor{movieName='movie03', actorName='actor2', actingAward='actingaward2',
  actingYear=2003, actingSchoolName='actingschool2'}
MovieActor{movieName='movie03', actorName='actor3', actingAward='actingaward3',
  actingYear=2003, actingSchoolName='null'}
MovieActor{movieName='movie03', actorName='actor4', actingAward='actingaward4',
  actingYear=2003, actingSchoolName='actingschool4'}

```

Le code suivant définit deux [QueryConditional](#) instances. `QueryConditional` fonctionnent avec des valeurs clés (la clé de partition seule ou en combinaison avec la clé de tri) et correspondent aux principales [expressions conditionnelles de l'API du service](#) DynamoDB. Après la ligne de commentaire 1, l'exemple définit l'`keyEqual` instance qui correspond aux éléments dont la valeur de partition est de **movie01**.

Cet exemple définit également une expression de filtre qui filtre tout élément non **actingschoolname** activé après la ligne de commentaire 2.

Après la ligne de commentaire 3, l'exemple montre l'[QueryEnhancedRequest](#) instance que le code transmet à la `DynamoDbTable.query()` méthode. Cet objet combine la condition clé et le filtre utilisés par le SDK pour générer la demande au service DynamoDB.

```

public static void query(DynamoDbTable movieActorTable) {

    // 1. Define a QueryConditional instance to return items matching a partition
    value.
    QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->
b.partitionValue("movie01"));
    // 1a. Define a QueryConditional that adds a sort key criteria to the partition
    value criteria.

```

```

    QueryConditional sortGreaterThanOrEqualTo =
QueryConditional.sortGreaterThanOrEqualTo(b ->
b.partitionValue("movie01").sortValue("actor2"));
    // 2. Define a filter expression that filters out items whose attribute value
is null.
    final Expression filterOutNoActingschoolname =
Expression.builder().expression("attribute_exists(actingschoolname)").build();

    // 3. Build the query request.
QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(keyEqual)
    .filterExpression(filterOutNoActingschoolname)
    .build();
    // 4. Perform the query.
PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);
logger.info("page count: {}", pagedResults.stream().count()); // Log number of
pages.

pagedResults.items().stream()
    .sorted()
    .forEach(
        item -> logger.info(item.toString()) // Log the sorted list of
items.
    );

```

Voici le résultat de l'exécution de la méthode. La sortie affiche les éléments dont `movieName` la valeur est `movie01` et n'affiche aucun élément dont la valeur est `actingSchoolName` égale à **null**

```

2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
actingYear=2001, actingSchoolName='actingschool1'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
actingYear=2001, actingSchoolName='actingschool4'}

```

Dans la variante de demande de requête suivante présentée précédemment après la ligne de commentaire 3, le code remplace le `keyEqual QueryConditional` par

`sortGreaterThanOrEqualTo` `QueryConditional` celui qui a été défini après la ligne de commentaire 1a. Le code suivant supprime également l'expression du filtre.

```
QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(sortGreaterThanOrEqualTo)
```

Comme cette table possède une clé primaire composite, toutes les `QueryConditional` instances nécessitent une valeur de clé de partition. `QueryConditional`les méthodes qui commencent par `sort...` indiquent qu'une clé de tri est requise. Les résultats ne sont pas triés.

La sortie suivante affiche les résultats de la requête. La requête renvoie les éléments dont **movieName** la valeur est égale à `movie01` et uniquement les éléments dont **actorName** la valeur est supérieure ou égale à `actor2`. Le filtre ayant été supprimé, la requête renvoie des éléments qui n'ont aucune valeur pour l'`actingSchoolName`attribut.

```
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
```

Effectuer des opérations par lots

[L'API DynamoDB Enhanced Client propose deux méthodes par lots `batchGetItem\(\)` et `batchWriteItem`](#)

Exemple de `batchGetItem()`

Avec `DynamoDbTable.batchGetItem()` cette méthode, vous pouvez récupérer jusqu'à 100 éléments individuels dans plusieurs tables dans une seule demande globale. L'exemple suivant utilise les classes de `MovieActor` données `Customer` et présentées précédemment.

Dans l'exemple qui suit les lignes 1 et 2, vous créez `ReadBatch` des objets que vous ajoutez ultérieurement en tant que paramètres à la `batchGetItem()` méthode après la ligne de commentaire 3. Le code situé après la ligne de commentaire 1 crée le lot à lire dans le `Customer` tableau. Le code situé après la ligne de commentaire 1a montre l'utilisation d'un

[GetItemEnhancedRequest](#) générateur qui prend des valeurs de clé primaire pour spécifier l'élément à lire. Contrairement à la spécification de valeurs clés pour demander un élément, vous pouvez utiliser une classe de données pour demander un élément, comme indiqué après la ligne de commentaire 1b. Le SDK extrait les valeurs clés en arrière-plan avant de soumettre la demande.

[Lorsque vous spécifiez l'élément à l'aide de l'approche basée sur les clés, comme indiqué dans les deux instructions situées après 2a, vous pouvez également spécifier que DynamoDB doit effectuer une lecture très cohérente.](#) Lorsque la `consistentRead()` méthode est utilisée, elle doit être utilisée sur tous les éléments demandés pour la même table.

Pour récupérer les éléments trouvés par DynamoDB, utilisez [resultsForTable\(\)](#) la méthode indiquée après la ligne de commentaire 4. Appelez la méthode pour chaque table lue dans la demande. `resultsForTable()` renvoie une liste des éléments trouvés que vous pouvez traiter à l'aide de n'importe quelle `java.util.List` méthode. Cet exemple enregistre chaque élément.

Pour découvrir les éléments que DynamoDB n'a pas traités, utilisez l'approche indiquée après la ligne de commentaire 5. La `BatchGetResultPage` classe possède la [unprocessedKeysForTable\(\)](#) méthode qui vous donne accès à chaque clé non traitée. La [référence de l'BatchGetItem API](#) contient plus d'informations sur les situations qui se traduisent par des éléments non traités.

```
public static void batchGetItemExample(DynamoDbEnhancedClient enhancedClient,
                                      DynamoDbTable<Customer> customerTable,
                                      DynamoDbTable<MovieActor> movieActorTable) {

    Customer customer2 = new Customer();
    customer2.setId("2");
    customer2.setEmail("cust2@example.org");

    // 1. Build a batch to read from the Customer table.
    ReadBatch customerBatch = ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        // 1a. Specify the primary key values for the item.
        .addGetItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
        // 1b. Alternatively, supply a data class instances to provide the
primary key values.
        .addGetItem(customer2)
        .build();

    // 2. Build a batch to read from the MovieActor table.
    ReadBatch movieActorBatch = ReadBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
```

```
        // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
table.
        .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor1")).consistentRead(Boolean.TRUE))
        .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor4")).consistentRead(Boolean.TRUE))
        .build();

    // 3. Add ReadBatch objects to the request.
    BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
b.readBatches(customerBatch, moveActorBatch));

    // 4. Retrieve the successfully requested items from each table.
    resultPages.resultsForTable(customerTable).forEach(item ->
logger.info(item.toString()));
    resultPages.resultsForTable(movieActorTable).forEach(item ->
logger.info(item.toString()));

    // 5. Retrieve the keys of the items requested but not processed by the
service.
    resultPages.forEach((BatchGetResultPage pageResult) -> {
        pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
        pageResult.unprocessedKeysForTable(movieActorTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
    });
}
```

Supposons que les éléments suivants se trouvent dans les deux tables avant d'exécuter l'exemple de code.

Éléments figurant dans les tableaux

```
Customer [id=1, name=CustName1, email=cust1@example.org,
regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
regDate=2023-03-31T15:46:28.688Z]
Customer [id=3, name=CustName3, email=cust3@example.org,
regDate=2023-03-31T15:46:29.688Z]
Customer [id=4, name=CustName4, email=cust4@example.org,
regDate=2023-03-31T15:46:30.688Z]
Customer [id=5, name=CustName5, email=cust5@example.org,
regDate=2023-03-31T15:46:31.689Z]
```

```

MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}

```

La sortie suivante montre les éléments renvoyés et enregistrés après la ligne de commentaire 4.

```

Customer [id=1, name=CustName1, email=cust1@example.org,
  regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
  regDate=2023-03-31T15:46:28.688Z]
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}

```

Exemple de `batchWriteItem()`

Le `batchWriteItem()` procédé place ou supprime plusieurs éléments dans une ou plusieurs tables. Vous pouvez spécifier jusqu'à 25 opérations de mise ou de suppression individuelles dans la demande. L'exemple suivant utilise les classes de [MovieActor](#) modèles [ProductCatalog](#)et présentées précédemment.

`WriteBatch`les objets sont créés après les lignes de commentaires 1 et 2. Pour le `ProductCatalog` tableau, le code place un élément et en supprime un. Pour le `MovieActor` tableau situé après la ligne de commentaire 2, le code place deux éléments et en supprime un.

La `batchWriteItem` méthode est appelée après la ligne de commentaire 3. Le [builder](#) paramètre fournit les demandes par lots pour chaque table.

L'[BatchWriteResult](#) objet renvoyé fournit des méthodes distinctes pour chaque opération afin de visualiser les demandes non traitées. Le code situé après la ligne de commentaire 4a fournit les clés pour les demandes de suppression non traitées et le code situé après la ligne de commentaire 4b fournit les éléments de saisie non traités.

```
public static void batchWriteItemExample(DynamoDbEnhancedClient enhancedClient,
```

```

                                DynamoDbTable<ProductCatalog>
catalogTable,
                                DynamoDbTable<MovieActor> movieActorTable)
{

    // 1. Build a batch to write to the ProductCatalog table.
    WriteBatch products = WriteBatch.builder(ProductCatalog.class)
        .mappedTableResource(catalogTable)
        .addPutItem(b -> b.item(getProductCatItem1()))
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getProductCatItem2().id())
            .sortValue(getProductCatItem2().title()))
        .build();

    // 2. Build a batch to write to the MovieActor table.
    WriteBatch movies = WriteBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
        .addPutItem(getMovieActorYeoh())
        .addPutItem(getMovieActorBlanchettPartial())
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getMovieActorStreep().getMovieName())
            .sortValue(getMovieActorStreep().getActorName()))
        .build();

    // 3. Add WriteBatch objects to the request.
    BatchWriteResult batchWriteResult = enhancedClient.batchWriteItem(b ->
b.writeBatches(products, movies));
    // 4. Retrieve keys for items the service did not process.
    // 4a. 'unprocessedDeleteItemsForTable()' returns keys for delete requests that
did not process.
    if (batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).size() >
0) {

batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).forEach(key ->
    logger.info(key.toString()));
    }
    // 4b. 'unprocessedPutItemsForTable()' returns keys for put requests that did
not process.
    if (batchWriteResult.unprocessedPutItemsForTable(catalogTable).size() > 0) {
        batchWriteResult.unprocessedPutItemsForTable(catalogTable).forEach(key ->
            logger.info(key.toString()));
    }
}
}

```

Les méthodes d'assistance suivantes fournissent les objets modèles pour les opérations de saisie et de suppression.

Méthodes auxiliaires

```
public static ProductCatalog getProductCatItem1() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatItem2() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchettPartial() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2023);
    movieActor.setActingAward("Best Actress");
    return movieActor;
}

public static MovieActor getMovieActorStreep() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Meryl Streep");
    movieActor.setMovieName("Sophie's Choice");
    movieActor.setActingYear(1982);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Yale School of Drama");
    return movieActor;
}

public static MovieActor getMovieActorYeoh(){
    MovieActor movieActor = new MovieActor();
```



```
movieActor.setActorName("Michelle Yeoh");
movieActor.setMovieName("Everything Everywhere All at Once");
movieActor.setActingYear(2023);
movieActor.setActingAward("Best Actress");
movieActor.setActingSchoolName("Royal Academy of Dance");
return movieActor;
}
```

Supposons que les tables contiennent les éléments suivants avant d'exécuter l'exemple de code.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
```

Une fois l'exemple de code terminé, les tableaux contiennent les éléments suivants.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='null'}
MovieActor{movieName='Everything Everywhere All at Once', actorName='Michelle Yeoh', actingAward='Best Actress', actingYear=2023, actingSchoolName='Royal Academy of Dance'}
ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b], price=30.22}
```

Notez dans le `MovieActor` tableau que l'élément du `Blue Jasmine` film a été remplacé par l'élément utilisé dans la demande de vente acquise via la méthode `getMovieActorBlanchettPartial()` assistance. Si aucune valeur d'attribut Data Bean n'a été fournie, la valeur de la base de données est supprimée. C'est pourquoi le résultat `actingSchoolName` est nul pour l'élément `Blue Jasmine` vidéo.

Note

[Bien que la documentation de l'API suggère que des expressions de condition peuvent être utilisées et que les mesures de capacité consommée et de collecte peuvent être renvoyées avec des demandes d'envoi et de suppression individuelles, ce n'est pas le cas dans un scénario d'écriture par lots.](#) Pour améliorer les performances des opérations par lots, ces options individuelles sont ignorées.

Réaliser des opérations de transaction

L'API DynamoDB Enhanced Client fournit `transactGetItems()` les méthodes et `transactWriteItems()` Les méthodes de transaction du SDK for Java assurent l'atomicité, la cohérence, l'isolation et la durabilité (ACID) des tables DynamoDB, vous aidant ainsi à maintenir l'exactitude des données dans vos applications.

Exemple de `transactGetItems()`

La [transactGetItems\(\)](#) méthode accepte jusqu'à 100 demandes individuelles d'articles. Tous les éléments sont lus en une seule transaction atomique. Le guide du développeur Amazon DynamoDB contient des informations sur les [conditions à l'origine de l'échec transactGetItems\(\) d'une méthode, ainsi que sur le niveau d'isolation utilisé lorsque vous appelez transactGetItem\(\)](#)

Après la ligne de commentaire 1 dans l'exemple suivant, le code appelle la `transactGetItems()` méthode avec un [builder](#) paramètre. Le générateur [addGetItem\(\)](#) est invoqué trois fois avec un objet de données contenant les valeurs clés que le SDK utilisera pour générer la demande finale.

La demande renvoie une liste d'[Document](#) objets après la ligne de commentaire 2. La liste des documents renvoyée contient des instances de [document](#) non nulles contenant des données d'article dans le même ordre que celui demandé. La [Document.getItem\(MappedTableResource<T> mappedTableResource\)](#) méthode convertit un objet non typé en Document objet Java typé si des données d'élément ont été renvoyées, sinon la méthode renvoie null.

```
public static void transactGetItemsExample(DynamoDbEnhancedClient enhancedClient,
                                          DynamoDbTable<ProductCatalog>
catalogTable,
                                          DynamoDbTable<MovieActor>
movieActorTable) {

    // 1. Request three items from two tables using a builder.
    final List<Document> documents = enhancedClient.transactGetItems(b -> b
        .addGetItem(catalogTable,
Key.builder().partitionValue(2).sortValue("Title 55").build())
        .addGetItem(movieActorTable, Key.builder().partitionValue("Sophie's
Choice").sortValue("Meryl Streep").build())
        .addGetItem(movieActorTable, Key.builder().partitionValue("Blue
Jasmine").sortValue("Cate Blanchett").build())
        .build());

    // 2. A list of Document objects is returned in the same order as requested.
    ProductCatalog title55 = documents.get(0).getItem(catalogTable);
```

```

    if (title55 != null) {
        logger.info(title55.toString());
    }

    MovieActor sophiesChoice = documents.get(1).getItem(movieActorTable);
    if (sophiesChoice != null) {
        logger.info(sophiesChoice.toString());
    }

    // 3. The getItem() method returns null if the Document object contains no item
    from DynamoDB.
    MovieActor blueJasmine = documents.get(2).getItem(movieActorTable);
    if (blueJasmine != null) {
        logger.info(blueJasmine.toString());
    }
}

```

Les tables DynamoDB contiennent les éléments suivants avant l'exécution de l'exemple de code.

```

ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
  Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}

```

La sortie suivante est enregistrée. Si un article est demandé mais introuvable, il n'est pas retourné comme c'est le cas pour la demande pour le film nommé Blue Jasmine.

```

ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
  Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}

```

Exemples `transactWriteItems()`

Il [transactWriteItems\(\)](#) accepte jusqu'à 100 actions de mise, de mise à jour ou de suppression dans une seule transaction atomique sur plusieurs tables. Le guide du développeur Amazon DynamoDB contient des informations détaillées sur les restrictions et les conditions de défaillance du fonctionnement du service [DynamoDB sous-jacent](#).

Exemple de base

Dans l'exemple suivant, quatre opérations sont demandées pour deux tables. Les classes [ProductCatalog](#) de modèles correspondantes [MovieActor](#) ont été présentées précédemment.

Chacune des trois opérations possibles (put, update et delete) utilise un paramètre de demande dédié pour spécifier les détails.

Le code après la ligne de commentaire 1 montre la variante simple de la `addPutItem()` méthode. La méthode accepte un [MappedTableResource](#) objet et l'instance d'objet de données à placer. L'instruction située après la ligne de commentaire 2 indique la variante qui accepte une [TransactPutItemEnhancedRequest](#) instance. Cette variante vous permet d'ajouter d'autres options dans la demande, comme une expression conditionnelle. L'[exemple](#) suivant montre une expression de condition pour une opération individuelle.

Une opération de mise à jour est demandée après la ligne de commentaire 3.

[TransactUpdateItemEnhancedRequest](#) possède une `ignoreNulls()` méthode qui vous permet de configurer ce que le SDK fait avec `null` les valeurs de l'objet modèle. Si la `ignoreNulls()` méthode renvoie `true`, le SDK ne supprime pas les valeurs d'attribut de la table pour les attributs d'objets de données qui le sont `null`. Si la `ignoreNulls()` méthode renvoie `false`, le SDK demande au service DynamoDB de supprimer les attributs de l'élément de la table. La valeur par défaut pour `ignoreNulls` est `false`.

L'instruction située après la ligne de commentaire 4 montre la variante d'une demande de suppression qui prend un objet de données. Le client amélioré extrait les valeurs clés avant d'envoyer la demande finale.

```
public static void transactWriteItems(DynamoDbEnhancedClient enhancedClient,
                                     DynamoDbTable<ProductCatalog> catalogTable,
                                     DynamoDbTable<MovieActor> movieActorTable) {

    enhancedClient.transactWriteItems(b -> b
        // 1. Simplest variation of put item request.
        .addPutItem(catalogTable, getProductCatId2())
        // 2. Put item request variation that accommodates condition
expressions.
        .addPutItem(movieActorTable,
TransactPutItemEnhancedRequest.builder(MovieActor.class)
            .item(getMovieActorStreep())

        .conditionExpression(Expression.builder().expression("attribute_not_exists
(movie)").build())
            .build())
        // 3. Update request that does not remove attribute values on the table
if the data object's value is null.
```

```

        .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId4ForUpdate())
            .ignoreNulls(Boolean.TRUE)
            .build())
        // 4. Variation of delete request that accepts a data object. The key
values are extracted for the request.
        .addDeleteItem(movieActorTable, getMovieActorBlanchett())
    );
}

```

Les méthodes d'assistance suivantes fournissent les objets de données pour les add*Item paramètres.

Méthodes auxiliaires

```

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Tar");
    movieActor.setActingYear(2022);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}

```

```
public static MovieActor getMovieActorStreep() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Meryl Streep");
    movieActor.setMovieName("Sophie's Choice");
    movieActor.setActingYear(1982);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Yale School of Drama");
    return movieActor;
}
```

Les tables DynamoDB contiennent les éléments suivants avant l'exécution de l'exemple de code.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
  actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

Les éléments suivants figurent dans les tableaux une fois l'exécution du code terminée.

```
3 | ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b],
  price=30.22}
4 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=40.0}
5 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
  Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

L'élément de la ligne 2 a été supprimé et les lignes 3 et 5 indiquent les articles qui ont été placés. La ligne 4 montre la mise à jour de la ligne 1. La `price` valeur est la seule valeur qui a changé sur l'élément. Si la valeur renvoyée `ignoreNulls()` était fausse, la ligne 4 ressemblerait à la ligne suivante.

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

Exemple de vérification de l'état

L'exemple suivant montre l'utilisation d'un contrôle de condition. Un contrôle de condition est utilisé pour vérifier l'existence d'un article ou pour vérifier l'état d'attributs spécifiques d'un article dans la base de données. L'article enregistré lors du contrôle de condition ne peut pas être utilisé dans le cadre d'une autre opération de la transaction.

Note

Vous ne pouvez pas cibler le même élément avec plusieurs opérations contenues dans la même transaction. Par exemple, vous ne pouvez pas effectuer de vérification de condition et essayer de mettre à jour le même article dans le cadre de la même transaction.

L'exemple montre un type d'opération de chaque type dans une demande d'écriture d'éléments transactionnelle. Après la ligne de commentaire 2, la `addConditionCheck()` méthode fournit la condition qui fait échouer la transaction si le `conditionExpression` paramètre est évalué à `false`. L'expression de condition renvoyée par la méthode indiquée dans le bloc des méthodes d'assistance vérifie si l'année de récompense du film n'"Sophie" s'Choiceest pas égale à 1982. Si c'est le cas, l'expression est évaluée à `false` et la transaction échoue.

Ce guide aborde les [expressions](#) de manière approfondie dans une autre rubrique.

```
public static void conditionCheckFailExample(DynamoDbEnhancedClient enhancedClient,
                                             DynamoDbTable<ProductCatalog>
catalogTable,
                                             DynamoDbTable<MovieActor>
movieActorTable) {
    try {
        enhancedClient.transactWriteItems(b -> b
            // 1. Perform one of each type of operation with the next three
            methods.
                .addPutItem(catalogTable,
                    TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
                        .item(getProductCatId2()).build())
                .addUpdateItem(catalogTable,
                    TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                        .item(getProductCatId4ForUpdate())
                        .ignoreNulls(Boolean.TRUE).build())
                .addDeleteItem(movieActorTable,
                    TransactDeleteItemEnhancedRequest.builder()
                        .key(b1 -> b1

                .partitionValue(getMovieActorBlanchett().getMovieName())

                .sortValue(getMovieActorBlanchett().getActorName()).build())
```

```

        // 2. Add a condition check on a table item that is not involved in
        another operation in this request.
        .addConditionCheck(movieActorTable, ConditionCheck.builder()
            .conditionExpression(buildConditionCheckExpression())
            .key(k -> k
                .partitionValue("Sophie's Choice")
                .sortValue("Meryl Streep"))
        // 3. Specify the request to return existing values from
        the item if the condition evaluates to true.

        .returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
            .build())
        .build());
    // 4. Catch the exception if the transaction fails and log the information.
} catch (TransactionCanceledException ex) {
    ex.cancellationReasons().stream().forEach(cancellationReason -> {
        logger.info(cancellationReason.toString());
    });
}
}
}

```

Les méthodes d'assistance suivantes sont utilisées dans l'exemple de code précédent.

Méthodes auxiliaires

```

private static Expression buildConditionCheckExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(1982));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}
}

```



```

public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2013);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}

```

Les tables DynamoDB contiennent les éléments suivants avant l'exécution de l'exemple de code.

```

1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
3 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}

```

Les éléments suivants figurent dans les tableaux une fois l'exécution du code terminée.

```

ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}

```

Les éléments restent inchangés dans les tables car la transaction a échoué. La `actingYear` valeur du film `Sophie's Choice` est 1982, comme indiqué à la ligne 2 des éléments du tableau avant l'appel de la `transactWriteItem()` méthode.

Pour saisir les informations d'annulation de la transaction, placez l'appel de `transactWriteItems()` méthode dans un `try` bloc et `catch` le

[TransactionCanceledException](#). Après la ligne de commentaire 4 de l'exemple, le code enregistre chaque [CancellationReason](#) objet. Étant donné que le code suivant la ligne de commentaire 3 de l'exemple indique que des valeurs doivent être renvoyées pour l'élément à l'origine de l'échec de la transaction, le journal affiche les valeurs de base de données brutes pour l'élément Sophie 's Choice vidéo.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Meryl Streep),
  movie=AttributeValue(S=Sophie's Choice), actingaward=AttributeValue(S=Best Actress),
  actingyear=AttributeValue(N=1982), actingschoolname=AttributeValue(S=Yale School of
  Drama)}, ~
  Code=ConditionalCheckFailed, Message=The conditional request failed.)
```

Exemple de condition de fonctionnement unique

L'exemple suivant montre l'utilisation d'une condition pour une seule opération dans une demande de transaction. L'opération de suppression après la ligne de commentaire 1 contient une condition qui vérifie la valeur de l'élément cible de l'opération par rapport à la base de données. Dans cet exemple, l'expression de condition créée avec la méthode d'assistance après la ligne de commentaire 2 indique que l'élément doit être supprimé de la base de données si l'année active du film n'est pas égale à 2013.

Les [expressions](#) sont abordées plus loin dans ce guide.

```
public static void singleOperationConditionFailExample(DynamoDbEnhancedClient
enhancedClient,
DynamoDbTable<ProductCatalog> catalogTable,
DynamoDbTable<MovieActor>
movieActorTable) {
    try {
        enhancedClient.transactWriteItems(b -> b
            .addPutItem(catalogTable,
TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
                .item(getProductCatId2())
                .build())
            .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                .item(getProductCatId4ForUpdate())
```

```

        .ignoreNulls(Boolean.TRUE).build())
    // 1. Delete operation that contains a condition expression
    .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
        .key((Key.Builder k) -> {
            MovieActor blanchett = getMovieActorBlanchett();
            k.partitionValue(blanchett.getMovieName())
                .sortValue(blanchett.getActorName());
        })
        .conditionExpression(buildDeleteItemExpression())

    .returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
        .build())
    .build());
    } catch (TransactionCanceledException ex) {
        ex.cancellationReasons().forEach(cancellationReason ->
logger.info(cancellationReason.toString()));
    }
}

// 2. Provide condition expression to check if 'actingyear' is not equal to 2013.
private static Expression buildDeleteItemExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(2013));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}
}

```

Les méthodes d'assistance suivantes sont utilisées dans l'exemple de code précédent.

Méthodes auxiliaires

```

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}
}

```

```

public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}
public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2013);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}

```

Les tables DynamoDB contiennent les éléments suivants avant l'exécution de l'exemple de code.

```

1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best
  Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}

```

Les éléments suivants figurent dans les tableaux une fois l'exécution du code terminée.

```

ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2023-03-15 11:29:07 [main] INFO org.example.tests.TransactDemoTest:168 -
  MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best
  Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}

```

Les éléments restent inchangés dans les tables car la transaction a échoué. La `actingYear` valeur du film `Blue Jasmine` est `2013` celle indiquée à la ligne 2 dans la liste des éléments avant l'exécution de l'exemple de code.

Les lignes suivantes sont enregistrées dans la console.

```

CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Cate Blanchett),
  movie=AttributeValue(S=Blue Jasmine), actingaward=AttributeValue(S=Best Actress),

```

```
actinyear=AttributeValue(N=2013), actingschoolname=AttributeValue(S=National
Institute of Dramatic Art)),
Code=ConditionalCheckFailed, Message=The conditional request failed)
```

Utiliser des indices secondaires

Les index secondaires améliorent l'accès aux données en définissant des clés alternatives que vous pouvez utiliser dans les opérations de requête et d'analyse. Les indices secondaires globaux (GSI) possèdent une clé de partition et une clé de tri qui peuvent être différentes de celles de la table de base. En revanche, les indices secondaires locaux (LSI) utilisent la clé de partition de l'index principal.

Annoter une classe de données avec des annotations d'index secondaires

Les attributs qui participent aux index secondaires nécessitent l'`@DynamoDbSecondarySortKey` annotation `@DynamoDbSecondaryPartitionKey` ou.

La classe suivante montre les annotations pour deux indices. Le GSI nommé `SubjectLastPostedDateIndex` utilise l'`Subject` attribut pour la clé de partition et `LastPostedDateTime` pour la clé de tri. Le LSI nommé `ForumLastPostedDateIndex` utilise le `ForumName` comme clé de partition et `LastPostedDateTime` comme clé de tri.

Notez que l'`Subject` attribut joue un double rôle. Il s'agit de la clé de tri de la clé primaire et de la clé de partition du GSI nommé `SubjectLastPostedDateIndex`.

classe **MessageThread**

La `MessageThread` classe peut être utilisée comme classe de données pour l'[exemple de table de threads figurant](#) dans le manuel Amazon DynamoDB Developer Guide.

Importations

```
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;
```

```
import java.util.List;
```

```
@DynamoDbBean
public class MessageThread {
    private String ForumName;
    private String Subject;
    private String Message;
    private String LastPostedBy;
    private String LastPostedDateTime;
    private Integer Views;
    private Integer Replies;
    private Integer Answered;
    private List<String> Tags;

    @DynamoDbPartitionKey
    public String getForumName() {
        return ForumName;
    }

    public void setForumName(String forumName) {
        ForumName = forumName;
    }

    // Sort key for primary index and partition key for GSI
    "SubjectLastPostedDateIndex".
    @DynamoDbSortKey
    @DynamoDbSecondaryPartitionKey(indexNames = "SubjectLastPostedDateIndex")
    public String getSubject() {
        return Subject;
    }

    public void setSubject(String subject) {
        Subject = subject;
    }

    // Sort key for GSI "SubjectLastPostedDateIndex" and sort key for LSI
    "ForumLastPostedDateIndex".
    @DynamoDbSecondarySortKey(indexNames = {"SubjectLastPostedDateIndex",
    "ForumLastPostedDateIndex"})
    public String getLastPostedDateTime() {
        return LastPostedDateTime;
    }
}
```

```
public void setLastPostedDateTime(String lastPostedDateTime) {
    LastPostedDateTime = lastPostedDateTime;
}
public String getMessage() {
    return Message;
}

public void setMessage(String message) {
    Message = message;
}

public String getLastPostedBy() {
    return LastPostedBy;
}

public void setLastPostedBy(String lastPostedBy) {
    LastPostedBy = lastPostedBy;
}

public Integer getViews() {
    return Views;
}

public void setViews(Integer views) {
    Views = views;
}

@DynamoDbSecondaryPartitionKey(indexNames = "ForumRepliesIndex")
public Integer getReplies() {
    return Replies;
}

public void setReplies(Integer replies) {
    Replies = replies;
}

public Integer getAnswered() {
    return Answered;
}

public void setAnswered(Integer answered) {
    Answered = answered;
}
```

```
public List<String> getTags() {
    return Tags;
}

public void setTags(List<String> tags) {
    Tags = tags;
}

public MessageThread() {
    this.Answered = 0;
    this.LastPostedBy = "";
    this.ForumName = "";
    this.Message = "";
    this.LastPostedDateTime = "";
    this.Replies = 0;
    this.Views = 0;
    this.Subject = "";
}

@Override
public String toString() {
    return "MessageThread{" +
        "ForumName='" + ForumName + '\'' +
        ", Subject='" + Subject + '\'' +
        ", Message='" + Message + '\'' +
        ", LastPostedBy='" + LastPostedBy + '\'' +
        ", LastPostedDateTime='" + LastPostedDateTime + '\'' +
        ", Views=" + Views +
        ", Replies=" + Replies +
        ", Answered=" + Answered +
        ", Tags=" + Tags +
        '}';
}
}
```

Création de l'index

À partir de la version 2.20.86 du SDK pour Java, la `createTable()` méthode génère automatiquement des index secondaires à partir des annotations de classes de données. Par défaut, tous les attributs de la table de base sont copiés dans un index et les valeurs de débit allouées sont de 20 unités de capacité de lecture et 20 unités de capacité d'écriture.

Toutefois, si vous utilisez une version du SDK antérieure à la version 2.20.86, vous devez créer l'index en même temps que le tableau, comme indiqué dans l'exemple suivant. Cet exemple crée les deux index de la Thread table. Le paramètre `builder` possède des méthodes pour configurer les deux types d'index, comme indiqué après les lignes de commentaire 1 et 2. Vous utilisez la `indexName()` méthode du générateur d'index pour associer les noms d'index spécifiés dans les annotations de classe de données au type d'index souhaité.

Ce code configure tous les attributs de table pour qu'ils apparaissent dans les deux index après les lignes de commentaires 3 et 4. De plus amples informations sur [les projections d'attributs](#) sont disponibles dans le manuel du développeur Amazon DynamoDB.

```
public static void createMessageThreadTable(DynamoDbTable<MessageThread>
messageThreadDynamoDbTable, DynamoDbClient dynamoDbClient) {
    messageThreadDynamoDbTable.createTable(b -> b
        // 1. Generate the GSI.
        .globalSecondaryIndices(gsi ->
gsi.indexName("SubjectLastPostedDateIndex")
        // 3. Populate the GSI with all attributes.
        .projection(p -> p
            .projectionType(ProjectionType.ALL))
        )
        // 2. Generate the LSI.
        .localSecondaryIndices(lsi -> lsi.indexName("ForumLastPostedDateIndex")
        // 4. Populate the LSI with all attributes.
        .projection(p -> p
            .projectionType(ProjectionType.ALL))
        )
    );
}
```

Requête à l'aide d'un index

L'exemple suivant interroge l'index secondaire local `ForumLastPostedDateIndex`.

Après la ligne de commentaire 2, vous créez un [QueryConditional](#) objet obligatoire lors de l'appel de la méthode [DynamoDbIndex.query\(\)](#).

Vous obtenez une référence à l'index que vous souhaitez interroger après la ligne de commentaire 3 en transmettant le nom de l'index. Après la ligne de commentaire 4, vous appelez la `query()` méthode sur l'index qui transmet l'`QueryConditional` objet.

Vous configurez également la requête pour renvoyer trois valeurs d'attribut, comme indiqué après la ligne de commentaire 5. Si elle n'`attributesToProject()` est pas appelée, la requête renvoie

toutes les valeurs d'attribut. Notez que les noms d'attributs spécifiés commencent par des lettres minuscules. Ces noms d'attributs correspondent à ceux utilisés dans la table, pas nécessairement aux noms d'attributs de la classe de données.

Après la ligne de commentaire 6, parcourez les résultats, enregistrez chaque élément renvoyé par la requête et stockez-le également dans la liste pour le renvoyer à l'appelant.

```
public static List<MessageThread> queryUsingSecondaryIndices(DynamoDbEnhancedClient
    enhancedClient,
                                                            String lastPostedDate,
                                                            DynamoDbTable<MessageThread> threadTable) {
    // 1. Log the parameter value.
    logger.info("lastPostedDate value: {}", lastPostedDate);

    // 2. Create a QueryConditional whose sort key value must be greater than or
    equal to the parameter value.
    QueryConditional queryConditional =
    QueryConditional.sortGreaterThanOrEqualTo(qc ->
        qc.partitionValue("Forum02").sortValue(lastPostedDate));

    // 3. Specify the index name to query the DynamoDbIndex instance.
    final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
    threadTable.index("ForumLastPostedDateIndex");

    // 4. Perform the query by using the QueryConditional object.
    final SdkIterable<Page<MessageThread>> pagedResult =
    forumLastPostedDateIndex.query(q -> q
        .queryConditional(queryConditional)
        // 5. Request three attribute in the results.
        .attributesToProject("forumName", "subject", "lastPostedDateTime"));

    List<MessageThread> collectedItems = new ArrayList<>();
    // 6. Iterate through the pages response and sort the items.
    pagedResult.stream().forEach(page -> page.items().stream()

    .sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))
        .forEach(mt -> {
            // 7. Log the returned items and add the collection to
            return to the caller.
            logger.info(mt.toString());
            collectedItems.add(mt);
        }));
}
```

```
    return collectedItems;
}
```

Les éléments suivants existent dans la base de données avant l'exécution de la requête.

```
MessageThread{ForumName='Forum01', Subject='Subject01', Message='Message01',
  LastPostedBy='', LastPostedDateTime='2023.03.28', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject02', Message='Message02',
  LastPostedBy='', LastPostedDateTime='2023.03.29', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject04', Message='Message04',
  LastPostedBy='', LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='Message08',
  LastPostedBy='', LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='Message10',
  LastPostedBy='', LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject03', Message='Message03',
  LastPostedBy='', LastPostedDateTime='2023.03.30', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject06', Message='Message06',
  LastPostedBy='', LastPostedDateTime='2023.04.02', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject09', Message='Message09',
  LastPostedBy='', LastPostedDateTime='2023.04.05', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum05', Subject='Subject05', Message='Message05',
  LastPostedBy='', LastPostedDateTime='2023.04.01', Views=0, Replies=0, Answered=0,
  Tags=null}
MessageThread{ForumName='Forum07', Subject='Subject07', Message='Message07',
  LastPostedBy='', LastPostedDateTime='2023.04.03', Views=0, Replies=0, Answered=0,
  Tags=null}
```

Les instructions de journalisation des lignes 1 et 6 génèrent la sortie de console suivante.

```
lastPostedDate value: 2023.03.31
MessageThread{ForumName='Forum02', Subject='Subject04', Message='', LastPostedBy='',
  LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='', LastPostedBy='',
  LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0, Tags=null}
```

```
MessageThread{ForumName='Forum02', Subject='Subject10', Message='', LastPostedBy='',  
LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0, Tags=null}
```

La requête a renvoyé des éléments avec une `forumName` valeur de `Forum02` et une `lastPostedDateTime` valeur supérieure ou égale à `2023.03.31`. Les résultats affichent message des valeurs avec une chaîne vide, bien que les message attributs contiennent des valeurs dans l'index. Cela est dû au fait que l'attribut message n'a pas été projeté après la ligne de commentaire 5.

Utiliser les fonctionnalités de cartographie avancées

Découvrez les fonctionnalités avancées du schéma de table dans l'API DynamoDB Enhanced Client.

Comprendre les types de schéma de table

[TableSchema](#) est l'interface de la fonctionnalité de mappage de l'API DynamoDB Enhanced Client. Il peut mapper un objet de données vers et depuis une carte de [AttributeValues](#). Un `TableSchema` objet doit connaître la structure de la table qu'il mappe. Ces informations de structure sont stockées dans un [TableMetadata](#) objet.

L'API client améliorée comporte plusieurs implémentations de `TableSchema`, qui suivent.

Schéma de table généré à partir de classes annotées

La création d'une classe à `TableSchema` partir de classes annotées est une opération modérément coûteuse. Nous vous recommandons donc de ne le faire qu'une seule fois, au démarrage de l'application.

[BeanTableSchema](#)

Cette implémentation est construite sur la base des attributs et des annotations d'une classe de bean. Un exemple de cette approche est présenté dans la [section Commencer](#).

Note

Si a ne `BeanTableSchema` se comporte pas comme prévu, activez la journalisation du débogage pour `software.amazon.awssdk.enhanced.dynamodb.beans`

[ImmutableTableSchema](#)

Cette implémentation est construite à partir d'une classe de données immuable. Cette approche est décrite dans la [???](#) section.

Schéma de table généré avec un générateur

Les éléments suivants `TableSchema` sont créés à partir du code à l'aide d'un générateur. Cette approche est moins coûteuse que celle qui utilise des classes de données annotées. L'approche du générateur évite l'utilisation d'annotations et ne nécessite pas de normes de JavaBean dénomination.

[StaticTableSchema](#)

Cette implémentation est conçue pour les classes de données mutables. La section de démarrage de ce guide explique comment [générer un `StaticTableSchema` l'aide d'un générateur](#).

[StaticImmutableTableSchema](#)

De la même manière que vous créez un `StaticTableSchema`, vous générez une implémentation de ce type en `TableSchema` utilisant un [générateur](#) à utiliser avec des classes de données immuables.

Schéma de table pour les données sans schéma fixe

[DocumentTableSchema](#)

Contrairement aux autres implémentations de `TableSchema`, vous ne définissez pas d'attributs pour une `DocumentTableSchema` instance. En général, vous ne spécifiez que les clés primaires et les fournisseurs de convertisseurs d'attributs. Une `EnhancedDocument` instance fournit les attributs que vous créez à partir d'éléments individuels ou d'une chaîne JSON.

Inclure ou exclure explicitement des attributs

L'API DynamoDB Enhanced Client propose des annotations pour empêcher les attributs de classe de données de devenir des attributs d'une table. Avec l'API, vous pouvez également utiliser un nom d'attribut différent du nom d'attribut de classe de données.

Exclure les attributs

Pour ignorer les attributs qui ne doivent pas être mappés à une table DynamoDB, marquez l'attribut avec l'annotation. `@DynamoDbIgnore`

```
private String internalKey;

@DynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
```

```
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

Inclure les attributs

Pour modifier le nom d'un attribut utilisé dans la table DynamoDB, marquez-le avec `@DynamoDbAttribute` l'annotation et saisissez un autre nom.

```
private String internalKey;

@dynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

Conversion des attributs de contrôle

Par défaut, un schéma de table fournit des convertisseurs pour de nombreux types Java courants via une implémentation par défaut de l'[AttributeConverterProvider](#) interface. Vous pouvez modifier le comportement général par défaut à l'aide d'une `AttributeConverterProvider` implémentation personnalisée. Vous pouvez également modifier le convertisseur pour un seul attribut.

Pour une liste des convertisseurs disponibles, consultez la documentation Java de [AttributeConverter](#) l'interface.

Fournir des fournisseurs de convertisseurs d'attributs personnalisés

Vous pouvez fournir un seul `AttributeConverterProvider` ou une chaîne de `AttributeConverterProvider` s ordonnés par le biais de l'`@DynamoDbBean(converterProviders = {...})` annotation. Toute personnalisation `AttributeConverterProvider` doit étendre l'`AttributeConverterProvider` interface.

Notez que si vous fournissez votre propre chaîne de fournisseurs de convertisseurs d'attributs, vous remplacerez le fournisseur de conversion par défaut, `DefaultAttributeConverterProvider`. Si vous souhaitez utiliser les fonctionnalités du `DefaultAttributeConverterProvider`, vous devez l'inclure dans la chaîne.

Il est également possible d'annoter le bean avec un tableau `{}` vide. Cela désactive l'utilisation de tous les fournisseurs de convertisseurs d'attributs, y compris le fournisseur par défaut. Dans ce cas, tous les attributs à mapper doivent disposer de leur propre convertisseur d'attributs.

L'extrait suivant montre un fournisseur de conversion unique.

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

L'extrait suivant montre l'utilisation d'une chaîne de fournisseurs de convertisseurs. La valeur par défaut du SDK étant fournie en dernier, elle a la priorité la plus basse.

```
@DynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
public class Customer {

}
```

Les constructeurs de schémas de tables statiques ont une `attributeConverterProviders()` méthode qui fonctionne de la même manière. Cela est illustré dans l'extrait suivant.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            a.getter(Customer::getName)
            a.setter(Customer::setName))
        .attributeConverterProviders(converterProvider1, converterProvider2)
        .build();
```

Remplacer le mappage d'un seul attribut

Pour modifier la façon dont un seul attribut est mappé, fournissez un `AttributeConverter` pour l'attribut. Cet ajout remplace tous les convertisseurs fournis `AttributeConverterProviders` dans le schéma de table. Cela ajoute un convertisseur personnalisé pour cet attribut uniquement. Les autres attributs, même ceux du même type, n'utiliseront pas ce convertisseur à moins qu'il ne soit explicitement spécifié pour ces autres attributs.

L'`@DynamoDbConvertedBy` annotation est utilisée pour spécifier la `AttributeConverter` classe personnalisée, comme indiqué dans l'extrait de code suivant.

```

@DynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(CustomAttributeConverter.class)
    public String getName() { return this.name; }
    public void setName(String name) { this.name = name;}
}

```

Les générateurs de schémas statiques utilisent une `attributeConverter()` méthode de génération d'attributs équivalente. Cette méthode prend une instance d'un, `AttributeConverter` comme le montre ce qui suit.

```

private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            a.getter(Customer::getName)
            a.setter(Customer::setName)
            a.attributeConverter(customAttributeConverter))
        .build();

```

Exemple

Cet exemple montre une `AttributeConverterProvider` implémentation qui fournit un convertisseur d'attributs pour les [java.net.HttpCookie](#) objets.

La `SimpleUser` classe suivante contient un attribut nommé `lastUsedCookie` qui est une instance de `HttpCookie`.

Le paramètre des `@DynamoDbBean` annotations répertorie les deux `AttributeConverterProvider` classes qui fournissent des convertisseurs.

Class with annotations

```

    @DynamoDbBean(converterProviders = {CookieConverterProvider.class,
    DefaultAttributeConverterProvider.class})
    public static final class SimpleUser {
        private String name;
        private HttpCookie lastUsedCookie;

        @DynamoDbPartitionKey

```



```

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public HttpCookie getLastUsedCookie() {
        return lastUsedCookie;
    }

    public void setLastUsedCookie(HttpCookie lastUsedCookie) {
        this.lastUsedCookie = lastUsedCookie;
    }

```

Static table schema

```

private static final TableSchema<SimpleUser> SIMPLE_USER_TABLE_SCHEMA =
    TableSchema.builder(SimpleUser.class)
        .newItemSupplier(SimpleUser::new)
        .attributeConverterProviders(CookieConverterProvider.create(),
AttributeConverterProvider.defaultProvider())
        .addAttribute(String.class, a -> a.name("name")
            .setter(SimpleUser::setName)
            .getter(SimpleUser::getName)
            .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(HttpCookie.class, a -> a.name("lastUsedCookie")
            .setter(SimpleUser::setLastUsedCookie)
            .getter(SimpleUser::getLastUsedCookie))
        .build();

```

CookieConverterProviderL'exemple suivant fournit une instance deHttpCookeConverter.

```

public static final class CookieConverterProvider implements
AttributeConverterProvider {
    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
ImmutableMap.of(
        // 1. Add HttpCookieConverter to the internal cache.
        EnhancedType.of(HttpCookie.class), new HttpCookieConverter());

    public static CookieConverterProvider create() {

```

```

        return new CookieConverterProvider();
    }

    // The SDK calls this method to find out if the provider contains a
    AttributeConverter instance
    // for the EnhancedType<T> argument.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }
}

```

Code de conversion

Dans la `transformFrom()` méthode de la `HttpCookieConverter` classe suivante, le code reçoit une `HttpCookie` instance et la transforme en une carte DynamoDB stockée sous forme d'attribut.

La `transformTo()` méthode reçoit un paramètre de carte DynamoDB, puis appelle le constructeur qui a besoin d'un `HttpCookie` nom et d'une valeur.

```

public static final class HttpCookieConverter implements
AttributeConverter<HttpCookie> {

    @Override
    public AttributeValue transformFrom(HttpCookie httpCookie) {

        return AttributeValue.fromM(
            Map.of ("cookieName", AttributeValue.fromS(httpCookie.getName()),
                "cookieValue", AttributeValue.fromS(httpCookie.getValue()))
        );
    }

    @Override
    public HttpCookie transformTo(AttributeValue attributeValue) {
        Map<String, AttributeValue> map = attributeValue.m();
        return new HttpCookie(
            map.get("cookieName").s(),
            map.get("cookieValue").s());
    }

    @Override
    public EnhancedType<HttpCookie> type() {

```

```

        return EnhancedType.of(HttpCookie.class);
    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.M;
    }
}

```

Modifier le comportement de mise à jour des attributs

Vous pouvez personnaliser le comportement de mise à jour des attributs individuels lorsque vous effectuez une opération de mise à jour. [UpdateItem \(\) et \(\) sont des exemples d'opérations de mise à jour dans l'API DynamoDB Enhanced Client. transactWriteItems](#)

Imaginons, par exemple, que vous souhaitez enregistrer un fichier créé sur horodatage dans votre dossier. Toutefois, vous souhaitez que sa valeur ne soit écrite que s'il n'existe aucune valeur existante pour l'attribut dans la base de données. Dans ce cas, vous utilisez le comportement de [WRITE_IF_NOT_EXISTS](#) mise à jour.

L'exemple suivant montre l'annotation qui ajoute le comportement à l'`createdOn` attribut.

```

@DynamoDbBean
public class Customer extends GenericRecord {
    private String id;
    private Instant createdOn;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }
    public void setId(String id) { this.name = id; }

    @DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
    public Instant getCreatedOn() { return this.createdOn; }
    public void setCreatedOn(Instant createdOn) { this.createdOn = createdOn; }
}

```

Vous pouvez déclarer le même comportement de mise à jour lorsque vous créez un schéma de table statique, comme indiqué dans l'exemple suivant après la ligne de commentaire 1.

```

static final TableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    TableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)

```

```

        .addAttribute(String.class, a -> a.name("id")
            .getter(Customer::getId)
            .setter(Customer::setId)

.tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(Instant.class, a -> a.name("createdOn")
            .getter(Customer::getCreatedOn)
            .setter(Customer::setCreatedOn)
            // 1. Add an UpdateBehavior.

.tags(StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)))
        .build();

```

Aplatir les attributs des autres classes

Si les attributs de votre table sont répartis dans plusieurs classes Java différentes, que ce soit par héritage ou par composition, l'API DynamoDB Enhanced Client permet de regrouper les attributs en une seule classe.

Utiliser l'héritage

Si vos classes utilisent l'héritage, appliquez les approches suivantes pour aplatir la hiérarchie.

Utiliser des haricots annotés

Pour l'approche d'annotation, les deux classes doivent porter l'`@DynamoDbBean` annotation et une classe doit porter une ou plusieurs annotations de clé primaire.

Vous trouverez ci-dessous des exemples de classes de données dotées d'une relation d'héritage.

Standard data class

```

@DynamoDbBean
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

@DynamoDbBean
public abstract class GenericRecord {
    private String id;

```

```

private String createdAt;

@DynamoDbPartitionKey
public String getId() { return id; }
public void setId(String id) { this.id = id; }

public String getCreatedAt() { return createdAt; }
public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

Lombok

L'[onMethodOption](#) de Lombok copie les annotations DynamoDB basées sur les attributs, telles que, sur le code généré. `@DynamoDbPartitionKey`

```

@DynamoDbBean
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord {
    private String name;
}

@Data
@DynamoDbBean
public abstract class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdAt;
}

```

Utiliser des schémas statiques

Pour l'approche du schéma statique, utilisez la `extend()` méthode du générateur pour réduire les attributs de la classe parent sur ceux de la classe enfant. Ceci est affiché après la ligne de commentaire 1 dans l'exemple suivant.

```

        StaticTableSchema<org.example.tests.model.inheritance.stat.GenericRecord>
GENERIC_RECORD_SCHEMA =

StaticTableSchema.builder(org.example.tests.model.inheritance.stat.GenericRecord.class)
        // The partition key will be inherited by the top level mapper.

```

```

        .addAttribute(String.class, a -> a.name("id"))

    .getter(org.example.tests.model.inheritance.stat.GenericRecord::getId)

    .setter(org.example.tests.model.inheritance.stat.GenericRecord::setId)
        .tags(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("created_date"))

    .getter(org.example.tests.model.inheritance.stat.GenericRecord::getCreatedDate)

    .setter(org.example.tests.model.inheritance.stat.GenericRecord::setCreatedDate))
        .build();

    StaticTableSchema<org.example.tests.model.inheritance.stat.Customer>
CUSTOMER_SCHEMA =

StaticTableSchema.builder(org.example.tests.model.inheritance.stat.Customer.class)

    .newItemSupplier(org.example.tests.model.inheritance.stat.Customer::new)
        .addAttribute(String.class, a -> a.name("name"))

    .getter(org.example.tests.model.inheritance.stat.Customer::getName)

    .setter(org.example.tests.model.inheritance.stat.Customer::setName))
        // 1. Use the extend() method to collapse the parent attributes
onto the child class.
        .extend(GENERIC_RECORD_SCHEMA) // All the attributes of the
GenericRecord schema are added to Customer.
        .build();

```

L'exemple de schéma statique précédent utilise les classes de données suivantes. Comme le mappage est défini lorsque vous créez le schéma de table statique, les classes de données ne nécessitent pas d'annotations.

Classes de données

Standard data class

```

public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

```

```

}

public abstract class GenericRecord {
    private String id;
    private String createdAt;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

Lombok

```

@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdAt;
}

```

Utiliser la composition

Si vos classes utilisent la composition, appliquez les approches suivantes pour aplatir la hiérarchie.

Utiliser des haricots annotés

L'`@DynamoDbFlat` annotation aplatit la classe contenue.

Les exemples de classes de données suivants utilisent l'`@DynamoDbFlat` annotation pour ajouter efficacement tous les attributs de la `GenericRecord` classe contenue à la `Customer` classe.

Standard data class

```
@DynamoDbBean
```

```

public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    @DynamoDbFlatten
    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

    @DynamoDbBean
    public class GenericRecord {
        private String id;
        private String createdAt;

        @DynamoDbPartitionKey
        public String getId() { return this.id; }
        public void setId(String id) { this.id = id; }

        public String getCreatedAt() { return this.createdAt; }
        public void setCreatedAt(String createdAt) { this.createdAt =
        createdAt; }
    }
}

```

Lombok

```

@Data
@DynamoDbBean
public class Customer {
    private String name;
    @Getter(onMethod_=@DynamoDbFlatten)
    private GenericRecord record;
}

@Data
@DynamoDbBean
public class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdAt;
}

```


Vous pouvez utiliser l'annotation `aplatis` pour aplatis autant de classes éligibles que nécessaire. Les contraintes suivantes s'appliquent :

- Tous les noms d'attributs doivent être uniques une fois aplatis.
- Il ne doit jamais y avoir plus d'une clé de partition, clé de tri ou nom de table.

Utiliser des schémas statiques

Lorsque vous créez un schéma de table statique, utilisez la `flatten()` méthode du générateur. Vous fournissez également les méthodes `getter` et `setter` qui identifient la classe contenue.

```
StaticTableSchema<GenericRecord> GENERIC_RECORD_SCHEMA =
    StaticTableSchema.builder(GenericRecord.class)
        .newItemSupplier(GenericRecord::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(GenericRecord::getId)
            .setter(GenericRecord::setId)
            .tags(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("created_date")
            .getter(GenericRecord::getCreatedDate)
            .setter(GenericRecord::setCreatedDate))
        .build();

StaticTableSchema<Customer> CUSTOMER_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            .getter(Customer::getName)
            .setter(Customer::setName))
        // Because we are flattening a component object, we supply a
getter and setter so the
        // mapper knows how to access it.
        .flatten(GENERIC_RECORD_SCHEMA, Customer::getRecord,
Customer::setRecord)
        .build();
```

L'exemple de schéma statique précédent utilise les classes de données suivantes.

Classes de données

Standard data class

```
public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

public class GenericRecord {
    private String id;
    private String createdAt;

    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return this.createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}
```

Lombok

```
@Data
public class Customer {
    private String name;
    private GenericRecord record;
}

@Data
public class GenericRecord {
    private String id;
    private String createdAt;
}
```

Vous pouvez utiliser le modèle Builder pour aplatir autant de classes éligibles que nécessaire.

Implications pour les autres codes

Lorsque vous utilisez l'`@DynamoDbFlatten` attribut (ou la méthode du `flatten()` générateur), l'élément de DynamoDB contient un attribut pour chaque attribut de l'objet composé. Il inclut également les attributs de l'objet composant.

En revanche, si vous annotez une classe de données avec une classe composée et que vous ne l'utilisez pas `@DynamoDbFlatten`, l'élément est enregistré avec l'objet composé en tant qu'attribut unique.

Par exemple, comparez la `Customer` classe indiquée dans l'[exemple d'aplatissement avec la composition avec](#) et sans aplatissement de l'attribut `record`. Vous pouvez visualiser la différence avec JSON, comme indiqué dans le tableau suivant.

Avec aplatissement	Sans aplatissement
3 attributs	2 attributs
<pre>{ "id": "1", "createdDate": "today", "name": "my name" }</pre>	<pre>{ "id": "1", "record": { "createdDate": "today", "name": "my name" } }</pre>

La différence devient importante si vous disposez d'un autre code accédant à la table DynamoDB qui devrait trouver certains attributs.

Utilisation d'attributs imbriqués

Un attribut imbriqué dans DynamoDB est intégré dans un autre attribut. Les éléments de liste et les entrées cartographiques en sont des exemples.

En Java, un attribut imbriqué DynamoDB correspond à un membre d'une classe qui est un `List` ou un `Map`. Il correspond également à une instance d'un type complexe, tel que `Address` ou `PhoneNumber`, tel qu'utilisé dans la `Person` classe suivante.

classe Person

```
@DynamoDbBean
public class Person {
    Integer id;
    String firstName;
    String lastName;
    Integer age;
    Map<String, Address> addresses;
    List<PhoneNumber> phoneNumbers;

    List<String> hobbies;

    @DynamoDbPartitionKey
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }
}
```

```
    }

    public Map<String, Address> getAddresses() {
        return addresses;
    }

    public void setAddresses(Map<String, Address> addresses) {
        this.addresses = addresses;
    }

    public List<PhoneNumber> getPhoneNumbers() {
        return phoneNumbers;
    }

    public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
        this.phoneNumbers = phoneNumbers;
    }

    public List<String> getHobbies() {
        return hobbies;
    }

    public void setHobbies(List<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
            "id=" + id +
            ", firstName='" + firstName + '\'' +
            ", lastName='" + lastName + '\'' +
            ", age=" + age +
            ", addresses=" + addresses +
            ", phoneNumbers=" + phoneNumbers +
            ", hobbies=" + hobbies +
            '}';
    }
}
```

classe **Address**

```
@DynamoDbBean
```

```
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
        return this.street;
    }

    public String getCity() {
        return this.city;
    }

    public String getState() {
        return this.state;
    }

    public String getZipCode() {
        return this.zipCode;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public void setState(String state) {
        this.state = state;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
    }
}
```

```
        if (o == null || getClass() != o.getClass()) return false;
        Address address = (Address) o;
        return Objects.equals(street, address.street) && Objects.equals(city,
address.city) && Objects.equals(state, address.state) && Objects.equals(zipCode,
address.zipCode);
    }

    @Override
    public int hashCode() {
        return Objects.hash(street, city, state, zipCode);
    }

    @Override
    public String toString() {
        return "Address{" +
            "street='" + street + '\'' +
            ", city='" + city + '\'' +
            ", state='" + state + '\'' +
            ", zipCode='" + zipCode + '\'' +
            '}';
    }
}
```

classe **PhoneNumber**

```
@DynamoDbBean
public class PhoneNumber {
    String type;
    String number;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
```

```
        this.number = number;
    }

    @Override
    public String toString() {
        return "PhoneNumber{" +
            "type='" + type + '\'' +
            ", number='" + number + '\'' +
            '}';
    }
}
```

Attributs imbriqués sur la carte

Utiliser des classes annotées

Vous pouvez enregistrer des attributs imbriqués pour des classes personnalisées en les annotant. La `Address` classe et la `PhoneNumber` classe affichées précédemment sont annotées uniquement avec l'`@DynamoDbBean` annotation. Lorsque l'API DynamoDB Enhanced Client crée le schéma de table pour la classe à `Person` l'aide de l'extrait suivant, l'API découvre l'utilisation des classes et `PhoneNumber` et crée les mappages `Address` correspondants pour fonctionner avec DynamoDB.

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

Utiliser des schémas imbriqués

L'autre approche consiste à utiliser des générateurs de schémas de tables statiques pour chacune des classes, comme indiqué dans le code suivant.

Les schémas de table des `PhoneNumber` classes `Address` et sont abstraits dans le sens où ils ne peuvent pas être utilisés avec une table DynamoDB. Cela est dû au fait qu'ils ne disposent pas de définitions pour la clé primaire. Ils sont toutefois utilisés comme schémas imbriqués dans le schéma de table de la `Person` classe.

Après les lignes de commentaire 1 et 2 de la définition de `PERSON_TABLE_SCHEMA`, vous pouvez voir le code qui utilise les schémas de table abstraits. L'utilisation de `documentOf` dans la `EnhanceType.documentOf(...)` méthode n'indique pas que la méthode renvoie un `EnhancedDocument` type d'API de document améliorée. Dans ce contexte, la `documentOf(...)` méthode renvoie un objet qui sait comment mapper son argument de classe vers et depuis les attributs de table DynamoDB en utilisant l'argument de schéma de table.

Code de schéma statique

```
// Abstract table schema that cannot be used to work with a DynamoDB table,  
// but can be used as a nested schema.  
public static final TableSchema<Address> TABLE_SCHEMA_ADDRESS =  
TableSchema.builder(Address.class)  
    .newItemSupplier(Address::new)  
    .addAttribute(String.class, a -> a.name("street")  
        .getter(Address::getStreet)  
        .setter(Address::setStreet))  
    .addAttribute(String.class, a -> a.name("city")  
        .getter(Address::getCity)  
        .setter(Address::setCity))  
    .addAttribute(String.class, a -> a.name("zipcode")  
        .getter(Address::getZipCode)  
        .setter(Address::setZipCode))  
    .addAttribute(String.class, a -> a.name("state")  
        .getter(Address::getState)  
        .setter(Address::setState))  
    .build();  
  
// Abstract table schema that cannot be used to work with a DynamoDB table,  
// but can be used as a nested schema.  
public static final TableSchema<PhoneNumber> TABLE_SCHEMA_PHONENUMBER =  
TableSchema.builder(PhoneNumber.class)  
    .newItemSupplier(PhoneNumber::new)  
    .addAttribute(String.class, a -> a.name("type")  
        .getter(PhoneNumber::getType)  
        .setter(PhoneNumber::setType))  
    .addAttribute(String.class, a -> a.name("number")  
        .getter(PhoneNumber::getNumber)  
        .setter(PhoneNumber::setNumber))  
    .build();  
  
// A static table schema that can be used with a DynamoDB table.  
// The table schema contains two nested schemas that are used to perform mapping  
to/from DynamoDB.  
public static final TableSchema<Person> PERSON_TABLE_SCHEMA =  
    TableSchema.builder(Person.class)  
        .newItemSupplier(Person::new)  
        .addAttribute(Integer.class, a -> a.name("id")  
            .getter(Person::getId)  
            .setter(Person::setId)  
            .addTag(StaticAttributeTags.primaryPartitionKey()))
```

```

        .addAttribute(String.class, a -> a.name("firstName")
            .getter(Person::getFirstName)
            .setter(Person::setFirstName))
        .addAttribute(String.class, a -> a.name("lastName")
            .getter(Person::getLastName)
            .setter(Person::setLastName))
        .addAttribute(Integer.class, a -> a.name("age")
            .getter(Person::getAge)
            .setter(Person::setAge))
        .addAttribute(EnhancedType.listOf(String.class), a ->
a.name("hobbies")
            .getter(Person::getHobbies)
            .setter(Person::setHobbies))
        .addAttribute(EnhancedType.mapOf(
            EnhancedType.of(String.class),
            // 1. Use mapping functionality of the Address table
schema.
                EnhancedType.documentOf(Address.class,
TABLE_SCHEMA_ADDRESS)), a -> a.name("addresses")
            .getter(Person::getAddresses)
            .setter(Person::setAddresses))
        .addAttribute(EnhancedType.listOf(
            // 2. Use mapping functionality of the PhoneNumber table
schema.
                EnhancedType.documentOf(PhoneNumber.class,
TABLE_SCHEMA_PHONENUMBER)), a -> a.name("phoneNumbers")
            .getter(Person::getPhoneNumbers)
            .setter(Person::setPhoneNumbers))
        .build();

```

Attributs imbriqués du projet

Pour les `scan()` méthodes `query()` et, vous pouvez spécifier les attributs que vous souhaitez voir renvoyés dans les résultats en utilisant des appels de méthode tels que `addNestedAttributeToProject()` et `attributesToProject()`. L'API DynamoDB Enhanced Client convertit les paramètres d'appel de méthode Java [en expressions de projection](#) avant l'envoi de la demande.

L'exemple suivant remplit le `Person` tableau avec deux éléments, puis exécute trois opérations de numérisation.

Le premier scan accède à tous les éléments du tableau afin de comparer les résultats aux autres opérations d'analyse.

Le second scan utilise la méthode du [addNestedAttributeToProject\(\)](#) générateur pour renvoyer uniquement la valeur de `street` l'attribut.

La troisième opération d'analyse utilise la méthode du [attributesToProject\(\)](#) générateur pour renvoyer les données de l'attribut de premier niveau, `hobbies`. Le type d'attribut de `hobbies` est une liste. Pour accéder à des éléments de liste individuels, effectuez une `get()` opération sur la liste.

```

        personDynamoDbTable = getDynamoDbEnhancedClient().table("Person",
PERSON_TABLE_SCHEMA);
        PersonUtils.createPersonTable(personDynamoDbTable, getDynamoDbClient());
        // Use a utility class to add items to the Person table.
        List<Person> personList = PersonUtils.getItemsForCount(2);
        // This utility method performs a put against DynamoDB to save the instances in
the list argument.
        PersonUtils.putCollection(getDynamoDbEnhancedClient(), personList,
personDynamoDbTable);

        // The first scan logs all items in the table to compare to the results of the
subsequent scans.
        final PageIterable<Person> allItems = personDynamoDbTable.scan();
        allItems.items().forEach(p ->
            // 1. Log what is in the table.
            logger.info(p.toString()));

        // Scan for nested attributes.
        PageIterable<Person> streetScanResult = personDynamoDbTable.scan(b -> b
            // Use the 'addNestedAttributeToProject()' or
'addNestedAttributesToProject()' to access data nested in maps in DynamoDB.
            .addNestedAttributeToProject(
                NestedAttributeName.create("addresses", "work", "street")
            ));

        streetScanResult.items().forEach(p ->
            //2. Log the results of requesting nested attributes.
            logger.info(p.toString()));

        // Scan for a top-level list attribute.
        PageIterable<Person> phoneNumbersScanResult = personDynamoDbTable.scan(b -> b
            // Use the 'attributesToProject()' method to access first-level
attributes.
            .attributesToProject("hobbies"));

        phoneNumbersScanResult.items().forEach((p) -> {

```

```

// 3. Log the results of the request for the 'hobbies' attribute.
logger.info(p.toString());
// To access an item in a list, first get the parent attribute, 'hobbies',
then access items in the list.
String hobby = p.getHobbies().get(1);
// 4. Log an item in the list.
logger.info(hobby);
});

```

```

// Logged results from comment line 1.
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,
addresses={work=Address{street='street 21', city='city 21', state='state 21',
zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',
zipCode='22222'}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},
PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]}
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,
addresses={work=Address{street='street 11', city='city 11', state='state 11',
zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',
zipCode='11111'}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},
PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]}

// Logged results from comment line 2.
Person{id=null, firstName='null', lastName='null', age=null,
addresses={work=Address{street='street 21', city='null', state='null',
zipCode='null'}}, phoneNumbers=null, hobbies=null}
Person{id=null, firstName='null', lastName='null', age=null,
addresses={work=Address{street='street 11', city='null', state='null',
zipCode='null'}}, phoneNumbers=null, hobbies=null}

// Logged results from comment lines 3 and 4.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
hobby 21
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
hobby 11

```

Note

Si la `attributesToProject()` méthode suit une autre méthode de création qui ajoute les attributs que vous souhaitez projeter, la liste des noms d'attributs fournie au `attributesToProject()` remplace tous les autres noms d'attributs.

Une analyse effectuée avec l'`ScanEnhancedRequest` instance figurant dans l'extrait suivant renvoie uniquement les données relatives aux loisirs.

```
ScanEnhancedRequest lastOverwrites = ScanEnhancedRequest.builder()
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
    .addAttributeToProject("firstName")
    // If the 'attributesToProject()' method follows other builder methods
    that add attributes for projection,
    // its list of attributes replace all previous attributes.
    .attributesToProject("hobbies")
    .build();
PageIterable<Person> hobbiesOnlyResult =
    personDynamoDbTable.scan(lastOverwrites);
hobbiesOnlyResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
```

L'extrait de code suivant utilise d'abord la `attributesToProject()` méthode. Cet ordre préserve tous les autres attributs demandés.

```
ScanEnhancedRequest attributesPreserved = ScanEnhancedRequest.builder()
    // Use 'attributesToProject()' first so that the method call does not
    replace all other attributes
    // that you want to project.
    .attributesToProject("firstName")
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
    .addAttributeToProject("hobbies")
    .build();
PageIterable<Person> allAttributesResult =
    personDynamoDbTable.scan(attributesPreserved);
allAttributesResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
```

```

Person{id=null, firstName='first name 2', lastName='null', age=null,
  addresses={work=Address{street='street 21', city='null', state='null',
  zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='first name 1', lastName='null', age=null,
  addresses={work=Address{street='street 11', city='null', state='null',
  zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 1, hobby 11]}

```

Préservez les objets vides avec `@DynamoDbPreserveEmptyObject`

Si vous enregistrez un bean dans Amazon DynamoDB avec des objets vides et que vous souhaitez que le SDK recrée les objets vides lors de leur extraction, annotez le getter du bean interne avec `@DynamoDbPreserveEmptyObject`

Pour illustrer le fonctionnement de l'annotation, l'exemple de code utilise les deux beans suivants.

Exemple de haricots

La classe de données suivante contient deux `InnerBean` champs. La méthode `getInnerBeanWithoutAnno()`, n'est pas annotée avec `@DynamoDbPreserveEmptyObject` La `getInnerBeanWithAnno()` méthode est annotée.

```

@dynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
    { this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbPreserveEmptyObject

```

```

    public InnerBean getInnerBeanWithAnno() { return innerBeanWithAnno; }
    public void setInnerBeanWithAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithAnno = innerBeanWithAnno; }

    @Override
    public String toString() {
        return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
            .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
            .add("innerBeanWithAnno=" + innerBeanWithAnno)
            .add("id=" + id + "")
            .add("name=" + name + "")
            .toString();
    }
}

```

Les instances de la `InnerBean` classe suivante sont des champs de `MyBean` et sont initialisées en tant qu'objets vides dans l'exemple de code.

```

@DynamoDbBean
public class InnerBean {

    private String innerBeanField;

    public String getInnerBeanField() {
        return innerBeanField;
    }

    public void setInnerBeanField(String innerBeanField) {
        this.innerBeanField = innerBeanField;
    }

    @Override
    public String toString() {
        return "InnerBean{" +
            "innerBeanField='" + innerBeanField + '\'' +
            '}';
    }
}

```

L'exemple de code suivant enregistre un `MyBean` objet avec des beans internes initialisés dans `DynamoDB`, puis récupère l'élément. La sortie enregistrée indique que le `innerBeanWithoutAnno` est pas initialisé, mais qu'il `innerBeanWithAnno` a été créé.

```

public MyBean preserveEmptyObjectAnnoUsingGetItemExample(DynamoDbTable<MyBean>
myBeanTable) {
    // Save an item to DynamoDB.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(new InnerBean()); // Instantiate the inner bean.
    bean.setInnerBeanWithAnno(new InnerBean());    // Instantiate the inner bean.
    myBeanTable.putItem(bean);

    GetItemEnhancedRequest request = GetItemEnhancedRequest.builder()
        .key(Key.builder().partitionValue("1").build())
        .build();
    MyBean myBean = myBeanTable.getItem(request);

    logger.info(myBean.toString());
    // Output 'MyBean[innerBeanWithoutAnno=null,
innerBeanWithAnno=InnerBean{innerBeanField='null'}, id='1', name='null']'.

    return myBean;
}

```

Schéma statique alternatif

Vous pouvez utiliser la `StaticTableSchema` version suivante des schémas de table à la place des annotations sur les beans.

```

public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
        StaticTableSchema.builder(InnerBean.class)
            .newItemSupplier(InnerBean::new)
            .addAttribute(String.class, a -> a.name("innerBeanField")
                .getter(InnerBean::getInnerBeanField)
                .setter(InnerBean::setInnerBeanField))
            .build();

    return StaticTableSchema.builder(MyBean.class)
        .newItemSupplier(MyBean::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(MyBean::getId)
            .setter(MyBean::setId)
            .addTag(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("name"))

```



```

        .getter(MyBean::getName)
        .setter(MyBean::setName))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema),
        a -> a.name("innerBean1")
            .getter(MyBean::getInnerBeanWithoutAnno)
            .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.preserveEmptyObject(true)),
        a -> a.name("innerBean2")
            .getter(MyBean::getInnerBeanWithAnno)
            .setter(MyBean::setInnerBeanWithAnno))
    .build();
}

```

Évitez de sauvegarder les attributs nuls des objets imbriqués

Vous pouvez ignorer les attributs nuls des objets imbriqués lorsque vous enregistrez un objet de classe de données dans DynamoDB en appliquant l'annotation. `@DynamoDbIgnoreNulls` En revanche, les attributs de niveau supérieur contenant des valeurs nulles ne sont jamais enregistrés dans la base de données.

Pour illustrer le fonctionnement de l'annotation, l'exemple de code utilise les deux beans suivants.

Exemple de haricots

La classe de données suivante contient deux `InnerBean` champs. La méthode `gettergetInnerBeanWithoutAnno()`, n'est pas annotée. La `getInnerBeanWithIgnoreNullsAnno()` méthode est annotée avec `@DynamoDbIgnoreNulls`.

```

@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithIgnoreNullsAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }
}

```

```

public String getName() { return name; }
public void setName(String name) { this.name = name; }

public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
{ this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

@DynamoDbIgnoreNulls
public InnerBean getInnerBeanWithIgnoreNullsAnno() { return
innerBeanWithIgnoreNullsAnno; }
public void setInnerBeanWithIgnoreNullsAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithIgnoreNullsAnno = innerBeanWithAnno; }

@Override
public String toString() {
    return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
        .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
        .add("innerBeanWithIgnoreNullsAnno=" + innerBeanWithIgnoreNullsAnno)
        .add("id='" + id + "'")
        .add("name='" + name + "'")
        .toString();
}
}

```

Les instances de la InnerBean classe suivante sont des champs de MyBean et sont utilisées dans l'exemple de code suivant.

```

@DynamoDbBean
public class InnerBean {

    private String innerBeanFieldString;
    private Integer innerBeanFieldInteger;

    public String getInnerBeanFieldString() { return innerBeanFieldString; }
    public void setInnerBeanFieldString(String innerBeanFieldString)
{ this.innerBeanFieldString = innerBeanFieldString; }

    public Integer getInnerBeanFieldInteger() { return innerBeanFieldInteger; }
    public void setInnerBeanFieldInteger(Integer innerBeanFieldInteger)
{ this.innerBeanFieldInteger = innerBeanFieldInteger; }

    @Override
    public String toString() {

```

```

        return new StringJoiner(", ", InnerBean.class.getSimpleName() + "[", "]")
            .add("innerBeanFieldString='" + innerBeanFieldString + "'")
            .add("innerBeanFieldInteger=" + innerBeanFieldInteger)
            .toString();
    }
}

```

L'exemple de code suivant crée un `InnerBean` objet et attribue une valeur à un seul de ses deux attributs.

```

public void ignoreNullsAnnoUsingPutItemExample(DynamoDbTable<MyBean> myBeanTable) {
    // Create an InnerBean object and give only one attribute a value.
    InnerBean innerBeanOneAttributeSet = new InnerBean();
    innerBeanOneAttributeSet.setInnerBeanFieldInteger(200);

    // Create a MyBean instance and use the same InnerBean instance both for
    attributes.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(innerBeanOneAttributeSet);
    bean.setInnerBeanWithIgnoreNullsAnno(innerBeanOneAttributeSet);

    Map<String, AttributeValue> itemMap = myBeanTable.tableSchema().itemToMap(bean,
true);
    logger.info(itemMap.toString());
    // Log the map that is sent to the database.
    //
    {innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)}),
id=AttributeValue(S=1),
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)}})}

    // Save the MyBean object to the table.
    myBeanTable.putItem(bean);
}

```

Pour visualiser les données de bas niveau envoyées à DynamoDB, le code enregistre la carte attributaire avant d'enregistrer l'objet. `MyBean`

La sortie enregistrée montre qu'elle `innerBeanWithIgnoreNullsAnno` produit un attribut,

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)})
```

L'annotation `innerBeanWithoutAnno` génère deux attributs. L'un des attributs a une valeur de 200 et l'autre est un attribut de valeur nulle.

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)})
```

Représentation JSON de la carte attributaire

La représentation JSON suivante permet de visualiser plus facilement les données enregistrées dans DynamoDB.

```
{
  "id": {
    "S": "1"
  },
  "innerBeanWithIgnoreNullsAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      }
    }
  },
  "innerBeanWithoutAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      },
      "innerBeanFieldString": {
        "NULL": true
      }
    }
  }
}
```

Schéma statique alternatif

Vous pouvez utiliser la `StaticTableSchema` version suivante des schémas de table pour mettre en place des annotations de classe de données.

```
public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
```

```
StaticTableSchema.builder(InnerBean.class)
    .newItemSupplier(InnerBean::new)
    .addAttribute(String.class, a -> a.name("innerBeanFieldString")
        .getter(InnerBean::getInnerBeanFieldString)
        .setter(InnerBean::setInnerBeanFieldString))
    .addAttribute(Integer.class, a -> a.name("innerBeanFieldInteger")
        .getter(InnerBean::getInnerBeanFieldInteger)
        .setter(InnerBean::setInnerBeanFieldInteger))
    .build();

return StaticTableSchema.builder(MyBean.class)
    .newItemSupplier(MyBean::new)
    .addAttribute(String.class, a -> a.name("id")
        .getter(MyBean::getId)
        .setter(MyBean::setId)
        .addTag(primaryPartitionKey()))
    .addAttribute(String.class, a -> a.name("name")
        .getter(MyBean::getName)
        .setter(MyBean::setName))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema),
        a -> a.name("innerBeanWithoutAnno")
            .getter(MyBean::getInnerBeanWithoutAnno)
            .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.ignoreNulls(true)),
        a -> a.name("innerBeanWithIgnoreNullsAnno")
            .getter(MyBean::getInnerBeanWithIgnoreNullsAnno)
            .setter(MyBean::setInnerBeanWithIgnoreNullsAnno))
    .build();
}
```

Travaillez avec des documents JSON avec l'API de document améliorée pour DynamoDB

L'[API de document améliorée](#) pour AWS SDK for Java 2.x est conçue pour fonctionner avec des données orientées document qui n'ont aucun schéma fixe. Toutefois, il vous permet également d'utiliser des classes personnalisées pour mapper des attributs individuels.

L'Enhanced Document API est le successeur de l'[API Document](#) de la AWS SDK for Java v1.x.

Table des matières

- [Commencez à utiliser l'API Enhanced Document](#)
 - [Créez un DocumentTableSchema et un DynamoDbTable](#)
- [Créez des documents améliorés](#)
 - [Construire à partir d'une chaîne JSON](#)
 - [Construisez à partir d'éléments individuels](#)
- [Effectuer des opérations CRUD](#)
- [Accédez aux attributs de document améliorés sous forme d'objets personnalisés](#)
- [Utiliser et EnhancedDocument sans DynamoDB](#)

Commencez à utiliser l'API Enhanced Document

L'API de document améliorée nécessite les mêmes [dépendances](#) que celles requises pour l'API client améliorée DynamoDB. Cela nécessite également une [DynamoDbEnhancedClientInstance](#), comme indiqué au début de cette rubrique.

Étant donné que l'API Enhanced Document a été publiée avec la version 2.20.3 du AWS SDK for Java 2.x, vous avez besoin de cette version ou d'une version ultérieure.

Créez un **DocumentTableSchema** et un **DynamoDbTable**

Pour appeler des commandes sur une table DynamoDB à l'aide de l'API Enhanced Document, associez la table à un objet de ressource < > [DynamoDbTable côté client EnhancedDocument](#).

La `table()` méthode du client amélioré crée une `DynamoDbTable<EnhancedDocument>` instance et nécessite des paramètres pour le nom de la table DynamoDB et un `DocumentTableSchema`

Le générateur d'un [DocumentTableSchema](#) nécessite une clé d'index principale et un ou plusieurs fournisseurs de convertisseurs d'attributs. La `AttributeConverterProvider.defaultProvider()` méthode fournit des convertisseurs pour les [types par défaut](#). Il doit être spécifié même si vous fournissez un fournisseur de convertisseur d'attributs personnalisé. Vous pouvez ajouter une clé d'index secondaire facultative au générateur.

L'extrait de code suivant montre le code qui génère la représentation côté client d'une table DynamoDB qui stocke des objets sans schéma. `person EnhancedDocument`

```
DynamoDbTable<EnhancedDocument> documentDynamoDbTable =  
    enhancedClient.table("person",
```

```
        TableSchema.documentSchemaBuilder()
            // Specify the primary key attributes.

        .addIndexPartitionKey(TableMetadata.primaryIndexName(),"id", AttributeValueType.S)
            .addIndexSortKey(TableMetadata.primaryIndexName(),
"lastName", AttributeValueType.S)
            // Specify attribute converter providers. Minimally add the
default one.

        .attributeConverterProviders(AttributeConverterProvider.defaultProvider())
            .build());

// Call documentTable.createTable() if "person" does not exist in DynamoDB.
// createTable() should be called only one time.
```

Ce qui suit montre la représentation JSON d'un person objet utilisé dans cette section.

personObjet JSON

```
{
  "id": 1,
  "firstName": "Richard",
  "lastName": "Roe",
  "age": 25,
  "addresses":
  {
    "home": {
      "zipCode": "00000",
      "city": "Any Town",
      "state": "FL",
      "street": "123 Any Street"
    },
    "work": {
      "zipCode": "00001",
      "city": "Anywhere",
      "state": "FL",
      "street": "100 Main Street"
    }
  },
  "hobbies": [
    "Hobby 1",
    "Hobby 2"
  ],
  "phoneNumbers": [
```

```
{
  "type": "Home",
  "number": "555-0100"
},
{
  "type": "Work",
  "number": "555-0119"
}
]
```

Créez des documents améliorés

An [EnhancedDocument](#) représente un objet de type document doté d'une structure complexe avec des attributs imbriqués. An EnhancedDocument nécessite des attributs de haut niveau qui correspondent aux attributs de clé primaire spécifiés pour leDocumentTableSchema. Le contenu restant est arbitraire et peut être composé d'attributs de haut niveau ou d'attributs profondément imbriqués.

Vous créez une EnhancedDocument instance à l'aide d'un générateur qui propose plusieurs méthodes pour ajouter des éléments.

Construire à partir d'une chaîne JSON

Avec une chaîne JSON, vous pouvez créer un appel de méthode EnhancedDocument en un. L'extrait suivant crée une chaîne EnhancedDocument à partir d'une chaîne JSON renvoyée par la méthode d'jsonPerson() assistance. La jsonPerson() méthode renvoie la version de chaîne JSON de l'[objet person](#) affiché précédemment.

```
EnhancedDocument document =
    EnhancedDocument.builder()
        .json( jsonPerson() )
        .build();
```

Construisez à partir d'éléments individuels

Vous pouvez également créer une EnhancedDocument instance à partir de composants individuels à l'aide des méthodes de type sécurisé du générateur.

L'exemple suivant crée un document person amélioré similaire au document amélioré créé à partir de la chaîne JSON de l'exemple précédent.

/* Define the shape of an address map whose JSON representation looks like the following.

Use 'addressMapEnhancedType' in the following EnhancedDocument.builder() to simplify the code.

```
"home": {
  "zipCode": "00000",
  "city": "Any Town",
  "state": "FL",
  "street": "123 Any Street"
}*/
```

```
EnhancedType<Map<String, String>> addressMapEnhancedType =
    EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class));
```

// Use the builder's typesafe methods to add elements to the enhanced document.

```
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
```

/* Add the map of addresses whose JSON representation looks like the following.

```
{
  "home": {
    "zipCode": "00000",
    "city": "Any Town",
    "state": "FL",
    "street": "123 Any Street"
  }
} */
    .putMap("addresses", getAddresses(), EnhancedType.of(String.class),
addressMapEnhancedType)
    .putList("hobbies", List.of("Theater", "Golf"),
EnhancedType.of(String.class))
    .build();
```

Méthodes auxiliaires

```
private static String phoneNumbersJSONString() {
```

```

        return " [" +
            " {" +
            "   \"type\": \"Home\", " +
            "   \"number\": \"555-0140\"" +
            " }, " +
            " {" +
            "   \"type\": \"Work\", " +
            "   \"number\": \"555-0155\"" +
            " }" +
            " ]";
    }

    private static Map<String, Map<String, String>> getAddresses() {
        return Map.of(
            "home", Map.of(
                "zipCode", "00002",
                "city", "Any Town",
                "state", "ME",
                "street", "123 Any Street"));
    }
}

```

Effectuer des opérations CRUD

Après avoir défini une `EnhancedDocument` instance, vous pouvez l'enregistrer dans une table DynamoDB. L'extrait de code suivant utilise le [PersonDocument](#) créé à partir d'éléments individuels.

```
documentDynamoDbTable.putItem(personDocument);
```

Après avoir lu une instance de document améliorée depuis DynamoDB, vous pouvez extraire les valeurs d'attribut individuelles à l'aide de getters, comme indiqué dans l'extrait de code suivant qui accède aux données enregistrées depuis le `personDocument`. Vous pouvez également extraire le contenu complet dans une chaîne JSON, comme indiqué dans la dernière partie de l'exemple de code.

```

// Read the item.
EnhancedDocument personDocFromDb =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).build());

// Access top-level attributes.
logger.info("Name: {} {}", personDocFromDb.getString("firstName"),
personDocFromDb.getString("lastName"));

```

```

// Name: Shirley Rodriguez

// Typesafe access of a deeply nested attribute. The addressMapEnhancedType
shown previously defines the shape of an addresses map.
Map<String, Map<String, String>> addresses =
personDocFromDb.getMap("addresses", EnhancedType.of(String.class),
addressMapEnhancedType);
addresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));
// {zipCode=00002, city=Any Town, street=123 Any Street, state=ME}

// Alternatively, work with AttributeValue types checking along the way for
deeply nested attributes.
Map<String, AttributeValue> addressesMap =
personDocFromDb.getMapOfUnknownType("addresses");
addressesMap.keySet().forEach((String k) -> {
    logger.info("Looking at data for [{}] address", k);
    // Looking at data for [home] address
    AttributeValue value = addressesMap.get(k);
    AttributeValue cityValue = value.m().get("city");
    if (cityValue != null) {
        logger.info(cityValue.s());
        // Any Town
    }
});

List<AttributeValue> phoneNumbers =
personDocFromDb.getListOfUnknownType("phoneNumbers");
phoneNumbers.forEach((AttributeValue av) -> {
    if (av.hasM()) {
        AttributeValue type = av.m().get("type");
        if (type.s() != null) {
            logger.info("Type of phone: {}", type.s());
            // Type of phone: Home
            // Type of phone: Work
        }
    }
});

String jsonPerson = personDocFromDb.toJson();
logger.info(jsonPerson);
// {"firstName":"Shirley","lastName":"Rodriguez","addresses":
{"home":{"zipCode":"00002","city":"Any Town","street":"123 Any
Street","state":"ME"}}, "hobbies":["Theater","Golf"],

```

```
//      "id":50,"nullAttribute":null,"age":53,"phoneNumbers":
[{"number":"555-0140","type":"Home"},{"number":"555-0155","type":"Work"}]}
```

EnhancedDocument les instances peuvent être utilisées avec n'importe quelle méthode [DynamoDbTable](#) ou [DynamoDbEnhancedClient](#) à la place de classes de données mappées.

Accédez aux attributs de document améliorés sous forme d'objets personnalisés

En plus de fournir une API pour lire et écrire des attributs avec des structures sans schéma, l'API Enhanced Document vous permet de convertir des attributs depuis et vers des instances de classes personnalisées.

L'API de document améliorée utilise AttributeConverterProvider les valeurs s et AttributeConverter s affichées dans la section de [conversion des attributs de contrôle](#) dans le cadre de l'API client améliorée DynamoDB.

Dans l'exemple suivant, nous utilisons a CustomAttributeConverterProvider avec sa AddressConverter classe imbriquée pour convertir Address des objets.

Cet exemple montre que vous pouvez mélanger des données provenant de classes et des données provenant de structures créées selon les besoins. Cet exemple montre également que les classes personnalisées peuvent être utilisées à n'importe quel niveau d'une structure imbriquée. Les Address objets de cet exemple sont des valeurs utilisées dans une carte.

```
public static void attributeToAddressClassMappingExample(DynamoDbEnhancedClient
enhancedClient, DynamoDbClient standardClient) {
    String tableName = "customer";

    // Define the DynamoDbTable for an enhanced document.
    // The schema builder provides methods for attribute converter providers and
keys.
    DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
enhancedClient.table(tableName,
        DocumentTableSchema.builder()
            // Add the CustomAttributeConverterProvider along with the
default when you build the table schema.
            .attributeConverterProviders(
                List.of(
                    new CustomAttributeConverterProvider(),
                    AttributeConverterProvider.defaultProvider()))
            .addIndexPartitionKey(TableMetadata.primaryIndexName(), "id",
AttributeValueType.N)
```

```

        .addIndexSortKey(TableMetadata.primaryIndexName(), "lastName",
AttributeValueType.S)
        .build());
// Create the DynamoDB table if needed.
documentDynamoDbTable.createTable();
waitForTableCreation(tableName, standardClient);

// The getAddressessForCustomMappingExample() helper method that provides
'addresses' shows the use of a custom Address class
// rather than using a Map<String, Map<String, String> to hold the address
data.
Map<String, Address> addresses = getAddressessForCustomMappingExample();

// Build an EnhancedDocument instance to save an item with a mix of structures
defined as needed and static classes.
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
    // Note the use of 'EnhancedType.of(Address.class)' instead of the more
generic
    // 'EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class))' that was used in a previous example.
    .putMap("addresses", addresses, EnhancedType.of(String.class),
EnhancedType.of(Address.class))
    .putList("hobbies", List.of("Hobby 1", "Hobby 2"),
EnhancedType.of(String.class))
    .build();
// Save the item to DynamoDB.
documentDynamoDbTable.putItem(personDocument);

// Retrieve the item just saved.
EnhancedDocument srPerson =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).sortValue("Rodriguez").build())

// Access the addresses attribute.
Map<String, Address> srAddresses = srPerson.get("addresses",
    EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(Address.class)));

```

```

        srAddresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));

        documentDynamoDbTable.deleteTable();

// The content logged to the console shows that the saved maps were converted to
// Address instances.
Address{street='123 Main Street', city='Any Town', state='NC', zipCode='00000'}
Address{street='100 Any Street', city='Any Town', state='NC', zipCode='00000'}

```

CustomAttributeConverterProvidercode

```

public class CustomAttributeConverterProvider implements AttributeConverterProvider {

    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
        ImmutableMap.of(
            // 1. Add AddressConverter to the internal cache.
            EnhancedType.of(Address.class), new AddressConverter());

    public static CustomAttributeConverterProvider create() {
        return new CustomAttributeConverterProvider();
    }

    // 2. The enhanced client queries the provider for attribute converters if it
    // encounters a type that it does not know how to convert.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }

    // 3. Custom attribute converter
    private class AddressConverter implements AttributeConverter<Address> {
        // 4. Transform an Address object into a DynamoDB map.
        @Override
        public AttributeValue transformFrom(Address address) {

            Map<String, AttributeValue> attributeValueMap = Map.of(
                "street", AttributeValue.fromS(address.getStreet()),
                "city", AttributeValue.fromS(address.getCity()),
                "state", AttributeValue.fromS(address.getState()),
                "zipCode", AttributeValue.fromS(address.getZipCode()));

            return AttributeValue.fromM(attributeValueMap);
        }
    }
}

```

```
    }

    // 5. Transform the DynamoDB map attribute to an Address object.
    @Override
    public Address transformTo(AttributeValue attributeValue) {
        Map<String, AttributeValue> m = attributeValue.m();
        Address address = new Address();
        address.setStreet(m.get("street").s());
        address.setCity(m.get("city").s());
        address.setState(m.get("state").s());
        address.setZipCode(m.get("zipCode").s());

        return address;
    }

    @Override
    public EnhancedType<Address> type() {
        return EnhancedType.of(Address.class);
    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.M;
    }
}
}
```

classe **Address**

```
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
        return this.street;
    }

    public String getCity() {
```

```
        return this.city;
    }

    public String getState() {
        return this.state;
    }

    public String getZipCode() {
        return this.zipCode;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public void setState(String state) {
        this.state = state;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }
}
```

Méthode d'assistance qui fournit des adresses

La méthode d'assistance suivante fournit la carte qui utilise des Address instances personnalisées pour les valeurs plutôt que des Map<String, String> instances génériques pour les valeurs.

```
private static Map<String, Address> getAddressesForCustomMappingExample() {
    Address homeAddress = new Address();
    homeAddress.setStreet("100 Any Street");
    homeAddress.setCity("Any Town");
    homeAddress.setState("NC");
    homeAddress.setZipCode("00000");

    Address workAddress = new Address();
    workAddress.setStreet("123 Main Street");
    workAddress.setCity("Any Town");
}
```



```
workAddress.setState("NC");
workAddress.setZipCode("00000");

return Map.of("home", homeAddress,
             "work", workAddress);
}
```

Utiliser et **EnhancedDocument** sans DynamoDB

Bien que vous utilisiez généralement une instance d'un `EnhancedDocument` pour lire et écrire des éléments DynamoDB de type document, elle peut également être utilisée indépendamment de DynamoDB.

Vous pouvez `EnhancedDocuments` les utiliser pour leur capacité à convertir des chaînes JSON ou des objets personnalisés en cartes de bas niveau, `AttributeValues` comme indiqué dans l'exemple suivant.

```
public static void conversionWithoutDynamoDbExample() {
    Address address = new Address();
    address.setCity("my city");
    address.setState("my state");
    address.setStreet("my street");
    address.setZipCode("00000");

    // Build an EnhancedDocument instance for its conversion functionality alone.
    EnhancedDocument addressEnhancedDoc = EnhancedDocument.builder()
        // Important: You must specify attribute converter providers when you
        // build an EnhancedDocument instance not used with a DynamoDB table.
        .attributeConverterProviders(new CustomAttributeConverterProvider(),
        DefaultAttributeConverterProvider.create())
        .put("addressDoc", address, Address.class)
        .build();

    // Convert address to a low-level item representation.
    final Map<String, AttributeValue> addressAsAttributeMap =
addressEnhancedDoc.getMapOfUnknownType("addressDoc");
    logger.info("addressAsAttributeMap: {}", addressAsAttributeMap.toString());

    // Convert address to a JSON string.
    String addressAsJsonString = addressEnhancedDoc.getJson("addressDoc");
    logger.info("addressAsJsonString: {}", addressAsJsonString);
    // Convert addressEnhancedDoc back to an Address instance.
```

```
        Address addressConverted = addressEnhancedDoc.get("addressDoc",
Address.class);
        logger.info("addressConverted: {}", addressConverted.toString());
    }

    /* Console output:
        addressAsAttributeMap: {zipCode=AttributeValue(S=00000),
state=AttributeValue(S=my state), street=AttributeValue(S=my street),
city=AttributeValue(S=my city)}
        addressAsJsonString: {"zipCode":"00000","state":"my state","street":"my
street","city":"my city"}
        addressConverted: Address{street='my street', city='my city', state='my
state', zipCode='00000'}
    */
```

Note

Lorsque vous utilisez un document amélioré indépendant d'une table DynamoDB, assurez-vous de définir explicitement les fournisseurs de convertisseurs d'attributs dans le générateur. En revanche, le schéma de table de documents fournit les fournisseurs de conversion lorsqu'un document amélioré est utilisé avec une table DynamoDB.

Utiliser des extensions

L'API DynamoDB Enhanced Client prend en charge les extensions de plug-in qui fournissent des fonctionnalités allant au-delà des opérations de mappage. Les extensions ont deux méthodes de crochet, `beforeWrite()` et `afterRead()`. `beforeWrite()` modifie une opération d'écriture avant qu'elle ne se produise, et la `afterRead()` méthode modifie les résultats d'une opération de lecture après qu'elle se produise. Étant donné que certaines opérations (telles que les mises à jour d'éléments) effectuent à la fois une écriture puis une lecture, les deux méthodes hook sont appelées.

Les extensions sont chargées dans l'ordre dans lequel elles sont spécifiées dans le générateur de clients amélioré. L'ordre de chargement peut être important car une extension peut agir sur des valeurs transformées par une extension précédente.

L'API client améliorée est fournie avec un ensemble d'extensions de plug-in situées dans le [extensions](#) package. Par défaut, le client amélioré charge le [VersionedRecordExtension](#) et le [AtomicCounterExtension](#). Vous pouvez modifier le comportement par défaut à l'aide du

générateur de clients Enhance et charger n'importe quelle extension. Vous pouvez également n'en spécifier aucune si vous ne souhaitez pas utiliser les extensions par défaut.

Si vous chargez vos propres extensions, le client amélioré ne charge aucune extension par défaut. Si vous souhaitez obtenir le comportement fourni par l'une ou l'autre des extensions par défaut, vous devez l'ajouter explicitement à la liste des extensions.

Dans l'exemple suivant, une extension personnalisée nommée `verifyChecksumExtension` est chargée d'après le `VersionedRecordExtension`, qui est généralement chargée par défaut elle-même. Le `n'AtomicCounterExtension` est pas chargé dans cet exemple.

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
    VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
        .dynamoDbClient(dynamoDbClient)
        .extensions(versionedRecordExtension,
            verifyChecksumExtension)
        .build();
```

VersionedRecordExtension

Le `VersionedRecordExtension` est chargé par défaut et incrémente et suit le numéro de version d'un élément au fur et à mesure que les éléments sont écrits dans la base de données. Une condition sera ajoutée à chaque écriture qui entraîne l'échec de l'écriture si le numéro de version de l'élément persistant réel ne correspond pas à la valeur lue pour la dernière fois par l'application. Ce comportement assure efficacement un verrouillage optimiste pour les mises à jour des articles. Si un autre processus met à jour un élément entre le moment où le premier processus l'a lu et celui où il écrit une mise à jour, l'écriture échouera.

Pour spécifier l'attribut à utiliser pour suivre le numéro de version de l'article, balisez un attribut numérique dans le schéma du tableau.

L'extrait suivant indique que l'`versionattribut` doit contenir le numéro de version de l'article.

```
@DynamoDbVersionAttribute
public Integer getVersion() {...};
public void setVersion(Integer version) {...};
```

L'approche équivalente du schéma de table statique est illustrée dans l'extrait suivant.

```

.addAttribute(Integer.class, a -> a.name("version")
                .getter(Customer::getVersion)
                .setter(Customer::setVersion)
                // Apply the 'version' tag to the attribute.

.tags(VersionedRecordExtension.AttributeTags.versionAttribute())

```

AtomicCounterExtension

Le `AtomicCounterExtension` est chargé par défaut et incrémente un attribut numérique balisé chaque fois qu'un enregistrement est écrit dans la base de données. Les valeurs de début et d'incrément peuvent être spécifiées. Si aucune valeur n'est spécifiée, la valeur de départ est définie sur 0 et la valeur de l'attribut augmente de 1.

Pour spécifier quel attribut est un compteur, balisez un attribut de type Long dans le schéma du tableau.

L'extrait suivant montre l'utilisation des valeurs de début et d'incrément par défaut pour l'attribut `counter`

```

@DynamoDbAtomicCounter
public Long getCounter() {...};
public void setCounter(Long counter) {...};

```

L'approche du schéma de table statique est illustrée dans l'extrait suivant. L'extension du compteur atomique utilise une valeur de départ de 10 et augmente la valeur de 5 à chaque fois que l'enregistrement est écrit.

```

.addAttribute(Integer.class, a -> a.name("counter")
                .getter(Customer::getCounter)
                .setter(Customer::setCounter)
                // Apply the 'atomicCounter' tag to the
attribute with start and increment values.
                .tags(StaticAttributeTags.atomicCounter(10L,
5L))

```

AutoGeneratedTimestampRecordExtension

Le met `AutoGeneratedTimestampRecordExtension` automatiquement à jour les attributs balisés de type [Instant](#) avec un horodatage actuel chaque fois que l'élément est écrit avec succès dans la base de données.

Cette extension n'est pas chargée par défaut. Par conséquent, vous devez le spécifier en tant qu'extension personnalisée lorsque vous créez le client amélioré, comme indiqué dans le premier exemple de cette rubrique.

Pour spécifier l'attribut à mettre à jour avec l'horodatage actuel, balisez-le dans le Instant schéma de la table.

L'lastUpdateattribut est la cible du comportement de l'extension dans l'extrait suivant. Notez l'exigence selon laquelle l'attribut doit être un Instant type.

```
@DynamoDbAutoGeneratedTimestampAttribute
public Instant getLastUpdate() {...}
public void setLastUpdate(Instant lastUpdate) {...}
```

L'approche équivalente du schéma de table statique est illustrée dans l'extrait suivant.

```
.addAttribute(Instant.class, a -> a.name("lastUpdate")
                .getter(Customer::getLastUpdate)
                .setter(Customer::setLastUpdate)
                // Applying the 'autoGeneratedTimestamp' tag to
the attribute.

.tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())
```

Extensions personnalisées

La classe d'extension personnalisée suivante montre une beforeWrite() méthode qui utilise une expression de mise à jour. Après la ligne de commentaire 2, nous créons un SetAction pour définir l'registrationDateattribut si l'élément de la base de données n'en possède pas déjà unregistrationDate. Chaque fois qu'un Customer objet est mis à jour, l'extension s'assure que a registrationDate est défini.

```
public final class CustomExtension implements DynamoDbEnhancedClientExtension {

    // 1. In a custom extension, use an UpdateExpression to define what action to take
before
    //    an item is updated.
    @Override
    public WriteModification beforeWrite(DynamoDbExtensionContext.BeforeWrite context)
    {
        if ( context.operationContext().tableName().equals("Customer")
```

```

        && context.operationName().equals(OperationName.UPDATE_ITEM)) {
            return WriteModification.builder()
                .updateExpression(createUpdateExpression())
                .build();
        }
        return WriteModification.builder().build(); // Return an "empty"
WriteModification instance if the extension should not be applied.
// In this case, if the code is
not updating an item on the Customer table.
    }

    private static UpdateExpression createUpdateExpression() {

        // 2. Use a SetAction, a subclass of UpdateAction, to provide the values in the
update.
        SetAction setAction =
            SetAction.builder()
                .path("registrationDate")
                .value("if_not_exists(registrationDate, :regValue)")
                .putExpressionValue(":regValue",
AttributeValue.fromS(Instant.now().toString()))
                .build();
        // 3. Build the UpdateExpression with one or more UpdateAction.
        return UpdateExpression.builder()
            .addAction(setAction)
            .build();
    }
}
}

```

Utiliser l'API client améliorée DynamoDB de manière asynchrone

Si votre application nécessite des appels asynchrones non bloquants à DynamoDB, vous pouvez utiliser le [DynamoDbEnhancedAsyncClient](#). Elle est similaire à l'implémentation synchrone, mais avec les principales différences suivantes :

1. Lorsque vous créez le `DynamoDbEnhancedAsyncClient`, vous devez fournir la version asynchrone du client standard `DynamoDbAsyncClient`, comme indiqué dans l'extrait de code suivant.

```

DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dynamoDbAsyncClient)

```

```
.build();
```

2. Les méthodes qui renvoient un seul objet de données renvoient une partie `CompletableFuture` du résultat au lieu du seul résultat. Votre application peut ensuite effectuer d'autres travaux sans avoir à bloquer le résultat. L'extrait suivant montre la méthode asynchrone `getItem()`.

```
CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join(); // Now block and wait for the result.
```

3. Les méthodes qui renvoient des listes de résultats paginées renvoient un [SdkPublisher](#) au lieu d'un [SdkIterable](#) que le synchronisme `DynamoDbEnhanceClient` renvoie pour les mêmes méthodes. Votre application peut ensuite abonner un gestionnaire à cet éditeur pour traiter les résultats de manière asynchrone sans avoir à les bloquer.

```
PagePublisher<Customer> results = customerDynamoDbTable.query(r ->
    r.queryConditional(keyEqualTo(k -> k.partitionValue("Smith"))));
results.subscribe(myCustomerResultsProcessor);
// Perform other work and let the processor handle the results asynchronously.
```

Pour un exemple plus complet d'utilisation du `SdkPublisher` API, consultez [l'exemple présenté](#) dans la section de ce guide consacrée à la `scan()` méthode asynchrone.

Annotations de classes de données

Le tableau suivant répertorie les annotations qui peuvent être utilisées sur les classes de données et fournit des liens vers des informations et des exemples contenus dans ce guide. Le tableau est trié par ordre alphabétique croissant par nom d'annotation.

Annotations de classes de données utilisées dans ce guide

Nom de l'annotation	L'annotation s'applique à ¹	Ce qu'il fait	Où cela est indiqué dans ce guide
DynamoDbAtomicCounter	attribut ²	Incrémente un attribut numérique balisé chaque fois qu'un enregistrement est	Présentation et discussion.

Nom de l'annotation	L'annotation s'applique à ¹	Ce qu'il fait	Où cela est indiqué dans ce guide
		écrit dans la base de données.	
DynamoDbAttribute	attribute	Définit ou renomme une propriété de bean mappée à un attribut de table DynamoDB.	<ul style="list-style-type: none"> • Discussion initiale. • Section de mise en route : voir Note. • Exemples de méthodes In Query en MovieActor classe.
DynamoDbAutoGeneratedTimestampAttribute	attribute	Met à jour un attribut balisé avec un horodatage actuel chaque fois que l'élément est écrit avec succès dans la base de données	Présentation et discussion.
DynamoDbBean	class	Marque une classe de données comme mappable à un schéma de table.	Première utilisation dans la classe Customer dans la section Commencer. Plusieurs utilisations apparaissent tout au long du guide.
DynamoDbConvertedBy	attribute	Associe une personnalisation <code>AttributeConverter</code> à l'attribut annoté.	Discussion initiale et exemple.

Nom de l'annotation	L'annotation s'applique à ¹	Ce qu'il fait	Où cela est indiqué dans ce guide
DynamoDbFlatten	attribute	Aplatit tous les attributs d'une classe de données DynamoDB distincte et les ajoute en tant qu'attributs de premier niveau à l'enregistrement lu et écrit dans la base de données.	<ul style="list-style-type: none"> • Discussion initiale. • Implications pour les autres codes.
DynamoDbIgnore	attribute	Il en résulte que l'attribut reste non mappé.	<ul style="list-style-type: none"> • Discussion initiale. • À utiliser en ProductCatalog classe.
DynamoDbIgnoreNulls	attribute	Empêche l'enregistrement des attributs nuls des DynamoDb objets imbriqués.	Discussion et exemples.
DynamoDbImmutable	class	Marque une classe de données immuable comme mappable à un schéma de table.	<ul style="list-style-type: none"> • Présentation de l'annotation. • À utiliser en ProductCatalog classe. • À utiliser avec Lombok.

Nom de l'annotation	L'annotation s'applique à ¹	Ce qu'il fait	Où cela est indiqué dans ce guide
DynamoDbPartitionKey	attribute	Marque un attribut comme clé de partition principale (clé de hachage) de la DynamoDb table.	<ul style="list-style-type: none"> • Utilisation initiale dans la classe Customer dans la section Commencer • Avec Lombok.
DynamoDbP reserveEmptyObject	attribute	Spécifie que si aucune donnée n'est présente pour l'objet mappé à l'attribut annoté, l'objet doit être initialisé avec tous les champs nuls.	Discussion et exemples.
DynamoDbS econdaryPartitionKey	attribute	Marque un attribut comme clé de partition pour un index secondaire global.	<ul style="list-style-type: none"> • Utilisation dans les index secondaires et exemple. • Dans les exemples de méthodes Query. • Dans l'exemple de Lombok • Avec des classes immuables.

Nom de l'annotation	L'annotation s'applique à ¹	Ce qu'il fait	Où cela est indiqué dans ce guide
DynamoDbSecondarySortKey	attribute	Marque un attribut comme clé de tri facultative pour un index secondaire global ou local.	<ul style="list-style-type: none"> • Utilisation dans les index secondaires et exemple. • Dans les exemples de méthodes Query. • Dans l'exemple de Lombok. • Avec des classes immuables.
DynamoDbSortKey	attribute	Marque un attribut comme clé de tri primaire facultative (clé de plage).	<ul style="list-style-type: none"> • Section de démarrage sur le cours destiné aux clients. • Avec des classes immuables. • Dans l'exemple de Lombok. • Dans les exemples de méthodes Query.
DynamoDbUpdateBehavior	attribute	Spécifie le comportement lorsque cet attribut est mis à jour dans le cadre d'une opération de « mise à jour » telle que UpdateItem.	Introduction et exemple
DynamoDbVersionAttribute	attribute	Incrémente le numéro de version d'un article.	Présentation et discussion.

¹ Vous pouvez appliquer des annotations au niveau de l'attribut au getter ou au setter, mais pas aux deux. Ce guide montre les annotations sur les getters.

² Le terme `property` est normalement utilisé pour une valeur encapsulée dans une classe de JavaBean données. Toutefois, ce guide utilise ce terme à la `attribute` place, par souci de cohérence avec la terminologie utilisée par DynamoDB.

Travaillez avec Amazon EC2

Cette section fournit des exemples de programmation [Amazon EC2](#) utilisant la version AWS SDK for Java 2.x.

Rubriques

- [Gérer les Amazon EC2 instances](#)
- [Zones d'utilisation Régions AWS et de disponibilité](#)
- [Travaillez avec des groupes de sécurité dans Amazon EC2](#)
- [Utiliser les métadonnées d'instance Amazon EC2](#)

Gérer les Amazon EC2 instances

Créer une instance

Créez une nouvelle Amazon EC2 instance en appelant la [runInstances](#) méthode [Ec2Client](#), en lui fournissant un [RunInstancesRequest](#) contenant l'[Amazon Machine Image \(AMI\)](#) à utiliser et un type d'[instance](#).

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static String createEC2Instance(Ec2Client ec2, String name, String amiId ) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceId = response.instances().get(0).instanceId();

    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceId)
        .tags(tag)
        .build();

    try {
        ec2.createTags(tagRequest);
        System.out.printf(
            "Successfully started EC2 Instance %s based on AMI %s",
            instanceId, amiId);

        return instanceId;

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

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

Démarrer une instance

Pour démarrer une Amazon EC2 instance, appelez la [startInstances](#) méthode Ec2Client, en lui fournissant un [StartInstancesRequest](#) contenant l'ID de l'instance à démarrer.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

Code

```
public static void startInstance(Ec2Client ec2, String instanceId) {

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.startInstances(request);
    System.out.printf("Successfully started instance %s", instanceId);
}
```

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

Arrêter une instance

Pour arrêter une Amazon EC2 instance, appelez la [stopInstances](#) méthode Ec2Client, en lui fournissant un [StopInstancesRequest](#) contenant l'ID de l'instance à arrêter.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

Code

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
```

```
StopInstancesRequest request = StopInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

ec2.stopInstances(request);
System.out.printf("Successfully stopped instance %s", instanceId);
}
```

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

Redémarrer une instance

Pour redémarrer une Amazon EC2 instance, appelez la [rebootInstances](#) méthode Ec2Client, en lui fournissant un [RebootInstancesRequest](#) contenant l'ID de l'instance à redémarrer.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.RebootInstancesRequest;
```

Code

```
public static void rebootEC2Instance(Ec2Client ec2, String instanceId) {

    try {
        RebootInstancesRequest request = RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        ec2.rebootInstances(request);
        System.out.printf(
            "Successfully rebooted instance %s", instanceId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Décrire des instances

Pour répertorier vos instances, créez une [DescribeInstancesRequest](#) et appelez la méthode `Ec2Client` [describeInstances](#). Il renverra un [DescribeInstancesResponse](#) objet que vous pourrez utiliser pour répertorier les Amazon EC2 instances de votre compte et de votre 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 `reservations` de la classe `DescribeInstancesResponse`, puis appeler `instances` sur chaque objet [Reservation](#) renvoyé.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void describeEC2Instances( Ec2Client ec2){

    String nextToken = null;

    try {

        do {
            DescribeInstancesRequest request =
DescribeInstancesRequest.builder().maxResults(6).nextToken(nextToken).build();
            DescribeInstancesResponse response = ec2.describeInstances(request);

            for (Reservation reservation : response.reservations()) {
                for (Instance instance : reservation.instances()) {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is "+ instance.imageId());
                    System.out.println("Instance type is "+
instance.instanceType());
                    System.out.println("Instance state name is "+
instance.state().name());
                }
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    } catch (Ec2Exception e) {
        // Handle exception
    }
}
```



```
        System.out.println("monitoring information is "+
instance.monitoring().state());

    }
}
    nextToken = response.nextToken();
} while (nextToken != null);

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

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

Surveiller 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 en savoir plus sur la surveillance des instances, consultez la section [Surveillance Amazon EC2](#) dans le guide de Amazon EC2 l'utilisateur pour les instances Linux.

Pour commencer à surveiller une instance, vous devez en créer un [MonitorInstancesRequest](#) avec l'ID de l'instance à surveiller et le transmettre à la méthode `Ec2Client.monitorInstances`.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

Code

```
public static void monitorInstance( Ec2Client ec2, String instanceId) {
```

```
MonitorInstancesRequest request = MonitorInstancesRequest.builder()
    .instanceIds(instanceId).build();

ec2.monitorInstances(request);
System.out.printf(
    "Successfully enabled monitoring for instance %s",
    instanceId);
}
```

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

Arrêter la surveillance d'une instance

Pour arrêter de surveiller une instance, créez une [UnmonitorInstancesRequest](#) avec l'ID de l'instance pour arrêter la surveillance et transmettez-la à la méthode `Ec2Client.unmonitorInstances`.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

Code

```
public static void unmonitorInstance(Ec2Client ec2, String instanceId) {
    UnmonitorInstancesRequest request = UnmonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.unmonitorInstances(request);

    System.out.printf(
        "Successfully disabled monitoring for instance %s",
        instanceId);
}
```

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

En savoir plus

- [RunInstances](#) dans la référence de Amazon EC2 l'API

- [DescribeInstances](#) dans la référence de Amazon EC2 l'API
- [StartInstances](#) dans la référence de Amazon EC2 l'API
- [StopInstances](#) dans la référence de Amazon EC2 l'API
- [RebootInstances](#) dans la référence de Amazon EC2 l'API
- [MonitorInstances](#) dans la référence de Amazon EC2 l'API
- [UnmonitorInstances](#) dans la référence de Amazon EC2 l'API

Zones d'utilisation Régions AWS et de disponibilité

Décrire des régions

Pour répertorier les régions disponibles sur votre compte, appelez la méthode `Ec2Client.describeRegions`. Elle renvoie un [DescribeRegionsResponse](#). Appelez la méthode `regions` de l'objet renvoyé pour obtenir une liste d'objets [Region](#) qui représentent chaque région.

Importations

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
try {
    DescribeRegionsResponse regionsResponse = ec2.describeRegions();
    regionsResponse.regions().forEach(region -> {
        System.out.printf(
            "Found Region %s with endpoint %s\n",
            region.regionName(),
            region.endpoint());
        System.out.println();
    });
}
```

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

Décrire les zones de disponibilité

Pour répertorier chaque zone de disponibilité disponible pour votre compte, appelez la méthode `Ec2Client.describeAvailabilityZones`. Elle renvoie un [DescribeAvailabilityZonesResponse](#). Appelez sa `availabilityZones` méthode pour obtenir une liste d'[AvailabilityZone](#) objets représentant chaque zone de disponibilité.

Importations

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

Créez le client Ec2.

```
software.amazon.awssdk.regions.Region region =
software.amazon.awssdk.regions.Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

Appelez ensuite `describeAvailabilityZones ()` et récupérez les résultats.

```
DescribeAvailabilityZonesResponse zonesResponse =
ec2.describeAvailabilityZones();
zonesResponse.availabilityZones().forEach(zone -> {
    System.out.printf(
        "Found Availability Zone %s with status %s in region %s%n",
        zone.zoneName(),
        zone.state(),
        zone.regionName()
    );
    System.out.println();
});
```

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

Décrire les comptes

Pour répertorier les informations relatives à EC2 concernant votre compte, appelez la méthode `Ec2Client.describeAccountAttributes`. Cette méthode renvoie un [DescribeAccountAttributesResponse](#) objet. Invoquez cette `describeAccountAttributes` méthode d'objets pour obtenir une liste d'[AccountAttribute](#) objets. Vous pouvez parcourir la liste pour récupérer un `AccountAttribute` objet.

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

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccount {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();
```

```
        describeEC2Account(ec2);
        System.out.print("Done");
        ec2.close();
    }

    public static void describeEC2Account(Ec2Client ec2) {
        try {
            DescribeAccountAttributesResponse accountResults =
ec2.describeAccountAttributes();
            accountResults.accountAttributes().forEach(attribute -> {
                System.out.print("\n The name of the attribute is " +
attribute.attributeName());
                attribute.attributeValues().forEach(
                    myValue -> System.out.print("\n The value of the attribute is "
+ myValue.attributeValue()));
            });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

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

En savoir plus

- [Régions et zones de disponibilité](#) dans le guide de Amazon EC2 l'utilisateur pour les instances Linux
- [DescribeRegions](#) dans la référence de Amazon EC2 l'API
- [DescribeAvailabilityZones](#) dans la référence de Amazon EC2 l'API

Travaillez avec des groupes de sécurité dans Amazon EC2

Création d'un groupe de sécurité

Pour créer un groupe de sécurité, appelez la `createSecurityGroup` méthode `Ec2Client` avec un [CreateSecurityGroupRequest](#) qui contient le nom de la clé.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

Code

```
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
        .groupName(groupName)
        .description(groupDesc)
        .vpcId(vpcId)
        .build();

        CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

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

Configurer un groupe de sécurité

Un groupe de sécurité peut contrôler à la fois le trafic entrant (entrée) et sortant (sortie) vers vos instances. Amazon EC2

Pour ajouter des règles d'entrée à votre groupe de sécurité, utilisez la `authorizeSecurityGroupIngress` méthode `Ec2Client`, en fournissant le nom du groupe de sécurité et les règles d'accès ([IpPermission](#)) que vous souhaitez lui attribuer dans un objet. [AuthorizeSecurityGroupIngressRequest](#) L'exemple suivant montre comment ajouter des autorisations IP à un groupe de sécurité.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

```
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

Code

Tout d'abord, créez un client Ec2

```
Region region = Region.US_WEST_2;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

Ensuite, utilisez la méthode `Ec2Client`, `authorizeSecurityGroupIngress`

```
IpRange ipRange = IpRange.builder()
    .cidrIp("0.0.0.0/0").build();

IpPermission ipPerm = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(80)
    .fromPort(80)
    .ipRanges(ipRange)
    .build();

IpPermission ipPerm2 = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(22)
    .fromPort(22)
    .ipRanges(ipRange)
    .build();

AuthorizeSecurityGroupIngressRequest authRequest =
    AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

AuthorizeSecurityGroupIngressResponse authResponse =
    ec2.authorizeSecurityGroupIngress(authRequest);

System.out.printf(
    "Successfully added ingress policy to Security Group %s",
```



```
        groupName);

        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Pour ajouter une règle de sortie au groupe de sécurité, fournissez des données similaires dans la [AuthorizeSecurityGroupEgressRequest](#) méthode `Ec2Client.authorizeSecurityGroupEgress`

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

Décrire des groupes de sécurité

Pour décrire vos groupes de sécurité ou obtenir des informations à leur sujet, appelez la méthode `Ec2Client.describeSecurityGroups`. Elle renvoie un [DescribeSecurityGroupsResponse](#) que vous pouvez utiliser pour accéder à la liste des groupes de sécurité en appelant sa `securityGroups` méthode, qui renvoie une liste d'[SecurityGroup](#) objets.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {

    try {
        DescribeSecurityGroupsRequest request =
            DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId).build();

        DescribeSecurityGroupsResponse response =
            ec2.describeSecurityGroups(request);
    }
```

```
for(SecurityGroup group : response.securityGroups()) {
    System.out.printf(
        "Found Security Group with id %s, " +
        "vpc id %s " +
        "and description %s",
        group.groupId(),
        group.vpcId(),
        group.description());
}
} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

Supprimer un groupe de sécurité

Pour supprimer un groupe de sécurité, appelez la `deleteSecurityGroup` méthode `Ec2Client` en lui transmettant un identifiant [DeleteSecurityGroupRequest](#) contenant l'ID du groupe de sécurité à supprimer.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {

    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf(
            "Successfully deleted Security Group with id %s", groupId);
    }
}
```

```
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

En savoir plus

- [Amazon EC2 Groupes de sécurité](#) dans le guide de Amazon EC2 l'utilisateur pour les instances Linux
- [Autorisez le 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

Utiliser les métadonnées d'instance Amazon EC2

Un client SDK Java pour le service de métadonnées d'instance Amazon EC2 (client de métadonnées) permet à vos applications d'accéder aux métadonnées sur leur instance EC2 locale. Le client de métadonnées fonctionne avec l'instance locale d'[IMDSv2](#) (Instance Metadata Service v2) et utilise des requêtes orientées session.

Deux classes de clients sont disponibles dans le SDK. Le mode synchrone [Ec2MetadataClient](#) est destiné aux opérations de blocage et le mode synchrone [Ec2MetadataAsyncClient](#) est destiné aux cas d'utilisation asynchrones et non bloquants.

Mise en route

Pour utiliser le client de métadonnées, ajoutez l'artefact `imds` Maven à votre projet. Vous avez également besoin de classes pour un [SdkHttpClient](#) (ou un [SdkAsyncHttpClient](#) pour le variant asynchrone) sur le classpath.

Le code XML Maven suivant présente des extraits de dépendance pour l'utilisation du synchrone [URLConnectionHttpClient](#) ainsi que la dépendance pour les clients de métadonnées.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>VERSION</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>imds</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>url-connection-client</artifactId>
  </dependency>
  <!-- other dependencies -->
</dependencies>
```

Recherchez la dernière version de l'artefact dans le [référentiel central de Maven](#).

Pour utiliser un client HTTP asynchrone, remplacez l'extrait de dépendance de l'`url-connection-client` artefact. Par exemple, l'extrait suivant introduit l'[NettyNioAsyncHttpClient](#) implémentation.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>netty-nio-client</artifactId>
</dependency>
```

Utiliser le client de métadonnées

Instancier un client de métadonnées

Vous pouvez instancier une instance d'un objet synchrone `Ec2MetadataClient` lorsqu'une seule implémentation de l'`SdkHttpClient` interface est présente sur le classpath. Pour ce faire, appelez la `Ec2MetadataClient#create()` méthode statique, comme illustré dans l'extrait suivant.

```
Ec2MetadataClient client = Ec2MetadataClient.create(); //  
'Ec2MetadataAsyncClient#create' is the asynchronous version.
```

Si votre application possède plusieurs implémentations de `ISdkHttpAsyncClient` or, vous devez spécifier une implémentation que le client de métadonnées doit utiliser, comme indiqué dans la [the section called “Client HTTP configurable”](#) section.

Note

Pour la plupart des clients de service, tels qu'Amazon S3, le SDK for Java ajoute automatiquement des implémentations de `ISdkHttpAsyncClient` or. Si votre client de métadonnées utilise la même implémentation, cela `Ec2MetadataClient#create()` fonctionnera. Si vous avez besoin d'une implémentation différente, vous devez la spécifier lors de la création du client de métadonnées.

Envoyer des demandes

Pour récupérer les métadonnées de l'instance, instanciez la `EC2MetadataClient` classe et appelez la `get` méthode avec un paramètre de chemin qui spécifie la [catégorie de métadonnées de l'instance](#).

L'exemple suivant imprime la valeur associée à `ami-id` clé sur la console.

```
Ec2MetadataClient client = Ec2MetadataClient.create();  
Ec2MetadataResponse response = client.get("/latest/meta-data/ami-id");  
System.out.println(response.asString());  
client.close(); // Closes the internal resources used by the Ec2MetadataClient class.
```

Si le chemin n'est pas valide, la `get` méthode génère une exception.

Réutilisez la même instance client pour plusieurs demandes, mais appelez `close` le client lorsqu'il n'est plus nécessaire de libérer des ressources. Une fois la méthode `close` appelée, l'instance client ne peut plus être utilisée.

Analyser les réponses

Les métadonnées de l'instance EC2 peuvent être sorties dans différents formats. Le texte brut et le JSON sont les formats les plus couramment utilisés. Les clients de métadonnées offrent des moyens de travailler avec ces formats.

Comme le montre l'exemple suivant, utilisez la `asString` méthode pour obtenir les données sous forme de chaîne Java. Vous pouvez également utiliser la `asList` méthode pour séparer une réponse en texte brut qui renvoie plusieurs lignes.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

Si la réponse est au format JSON, utilisez la `asDocument` méthode pour analyser la réponse JSON dans une instance de [Document](#), comme indiqué dans l'extrait de code suivant.

```
Document fullResponse = response.asDocument();
```

Une exception sera émise si le format des métadonnées n'est pas au format JSON. Si la réponse est analysée avec succès, vous pouvez utiliser l'[API du document](#) pour inspecter la réponse plus en détail. Consultez le [tableau des catégories de métadonnées](#) d'instance pour savoir quelles catégories de métadonnées fournissent des réponses au format JSON.

Configuration d'un client de métadonnées

Retries

Vous pouvez configurer un client de métadonnées avec un mécanisme de nouvelle tentative. Dans ce cas, le client peut réessayer automatiquement les demandes qui échouent pour des raisons inattendues. Par défaut, le client réessaie trois fois en cas d'échec d'une demande, avec un délai d'attente exponentiel entre les tentatives.

Si votre cas d'utilisation nécessite un mécanisme de nouvelle tentative différent, vous pouvez personnaliser le client à l'aide de la `retryPolicy` méthode de son générateur. Par exemple, l'exemple suivant montre un client synchrone configuré avec un délai fixe de deux secondes entre les tentatives et cinq nouvelles tentatives.

```
BackoffStrategy fixedBackoffStrategy =
    FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(retryPolicyBuilder ->
            retryPolicyBuilder.numRetries(5)

            .backoffStrategy(fixedBackoffStrategy))
        .build();
```

Il en existe plusieurs [BackoffStrategies](#) que vous pouvez utiliser avec un client de métadonnées.

Vous pouvez également désactiver complètement le mécanisme de nouvelle tentative, comme le montre l'extrait suivant.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(Ec2MetadataRetryPolicy.none())
        .build();
```

L'utilisation `Ec2MetadataRetryPolicy#none()` désactive la politique de nouvelle tentative par défaut afin que le client de métadonnées ne tente aucune nouvelle tentative.

Version IP

Par défaut, un client de métadonnées utilise le point de terminaison IPV4 sur `http://169.254.169.254`. Pour modifier le client afin qu'il utilise la version IPV6, utilisez `endpointMode` soit la `endpoint` méthode du générateur. Une exception se produit si les deux méthodes sont appelées sur le générateur.

Les exemples suivants illustrent les deux options IPV6.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .endpointMode(EndpointMode.IPV6)
        .build();
```

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .endpoint(URI.create("http://[fd00:ec2::254]"))
        .build();
```

Fonctions principales

Client asynchrone

Pour utiliser la version non bloquante du client, instanciez une instance de `laEc2MetadataAsyncClient` classe. Le code de l'exemple suivant crée un client asynchrone avec des paramètres par défaut et utilise `laget` méthode pour récupérer la valeur de `laami-id` clé.

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/ami-id");
```

Le résultat `java.util.concurrent.CompletableFuture` renvoyé par `laget` méthode se termine lorsque la réponse est renvoyée. L'exemple suivant imprime les `ami-id` métadonnées sur la console.

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

Client HTTP configurable

Le générateur de chaque client de métadonnées dispose d'une `httpClient` méthode que vous pouvez utiliser pour fournir un client HTTP personnalisé.

L'exemple suivant présente le code pour une `URLConnectionHttpClient` instance personnalisée.

```
SdkHttpClient httpClient =
    UrlConnectionHttpClient.builder()
        .socketTimeout(Duration.ofMinutes(5))
        .proxyConfiguration(proxy ->
            proxy.endpoint(URI.create("http://proxy.example.net:8888")))
        .build();
Ec2MetadataClient metaDataClient =
    Ec2MetadataClient.builder()
        .httpClient(httpClient)
        .build();
// Use the metaDataClient instance.
metaDataClient.close(); // Close the instance when no longer needed.
```

L'exemple suivant montre le code d'une `NettyNioAsyncHttpClient` instance personnalisée avec un client de métadonnées asynchrone.

```
SdkAsyncHttpClient httpAsyncClient =
```



```
NettyNioAsyncHttpClient.builder()
    .connectionTimeout(Duration.ofMinutes(5))
    .maxConcurrency(100)
    .build();
Ec2MetadataAsyncClient asyncMetaClient =
    Ec2MetadataAsyncClient.builder()
        .httpClient(httpAsyncClient)
        .build();
// Use the asyncMetaClient instance.
asyncMetaClient.close(); // Close the instance when no longer needed.
```

La [section called “Clients HTTP”](#) rubrique de ce guide explique en détail comment configurer les clients HTTP disponibles dans le SDK for Java.

Mise en cache de jetons

Comme les clients de métadonnées utilisent IMDSv2, toutes les demandes sont associées à une session. Une session est définie par un jeton qui a une date d'expiration, que le client de métadonnées gère pour vous. Chaque demande de métadonnées réutilise automatiquement le jeton jusqu'à son expiration.

Par défaut, un jeton dure six heures (21 600 secondes). Nous vous recommandons de conserver la time-to-live valeur par défaut, sauf si votre cas d'utilisation spécifique nécessite une configuration avancée.

Si nécessaire, configurez la durée à l'aide de la méthode `tokenTtl` générateur. Par exemple, le code de l'extrait suivant crée un client dont la durée de session est de cinq minutes.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .tokenTtl(Duration.ofMinutes(5))
        .build();
```

Si vous omettez d'appeler la `tokenTtl` méthode sur le générateur, la durée par défaut de 21 600 est utilisée à la place.

Travailler avec IAM

Cette section fournit des exemples de programmation AWS Identity and Access Management (IAM) à l'aide de la version AWS SDK for Java 2.x.

AWS Identity and Access Management(IAM) vous permet de contrôler en toute sécurité l'accès aux AWS services et aux ressources pour vos utilisateurs. À l'aide de IAM, vous pouvez créer et gérer des AWS utilisateurs et des groupes, et utiliser des autorisations pour autoriser ou refuser leur accès aux AWS ressources. Pour un guide complet IAM, consultez le [guide de l'IAM utilisateur](#).

Les exemples suivants 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

- [Gérer les clés IAM d'accès](#)
- [Gérer les IAM utilisateurs](#)
- [Créez des politiques IAM avec le AWS SDK for Java 2.x](#)
- [Travailler avec des IAM politiques](#)
- [Travailler avec des certificats IAM de serveur](#)

Gérer les clés IAM d'accès

Créer une clé d'accès

Pour créer une clé d' IAM accès, appelez la `IamClient`'s `createAccessKey` méthode avec un [CreateAccessKeyRequest](#) objet.

Note

Vous devez définir la région sur `AWS_GLOBAL` pour que les `IamClient` appels fonctionnent, car il s' IAM agit d'un service mondial.

Importations

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static String createIAMAccessKey(IamClient iam,String user) {

    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user).build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        String keyId = response.accessKey().accessKeyId();
        return keyId;

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

Répertoire des clés d'accès

Pour répertorier les clés d'accès d'un utilisateur donné, créez un [ListAccessKeysRequest](#) objet contenant le nom d'utilisateur pour lequel vous souhaitez répertorier les clés et transmettez-le à la `IamClient`'s `listAccessKeys` méthode.

Note

Si vous ne fournissez pas de nom d'utilisateur à `listAccessKeys`, il tentera de répertorier les clés d'accès associées au Compte AWS signataire de la demande.

Importations

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void listKeys( IamClient iam,String userName ){

    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName).build();
                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker).build();
                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata :
                response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                    metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Les résultats de `listAccessKeys` sont paginés par défaut (avec un maximum de 100 enregistrements par appel). Vous pouvez faire appel `isTruncated` à

L'[ListAccessKeysResponse](#) objet renvoyé pour voir si la requête a renvoyé moins de résultats que ceux disponibles. Si tel est le cas, appelez `marker` sur l'objet `ListAccessKeysResponse` et utilisez-la lors de la création d'une nouvelle demande. Utilisez cette nouvelle demande dans le prochain appel de `listAccessKeys`.

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

Extraire l'heure de la dernière utilisation d'une clé d'accès

Pour connaître l'heure à laquelle une clé d'accès a été utilisée pour la dernière fois, appelez la `IamClient`'s `getAccessKeyLastUsed` méthode avec l'ID de la clé d'accès (qui peut être transmis à l'aide d'un [GetAccessKeyLastUsedRequest](#) objet).

Vous pouvez ensuite utiliser l'[GetAccessKeyLastUsedResponse](#) objet renvoyé pour récupérer la date de dernière utilisation de la clé.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedRequest;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void getAccessKeyLastUsed(IamClient iam, String accessId ){

    try {
        GetAccessKeyLastUsedRequest request = GetAccessKeyLastUsedRequest.builder()
            .accessKeyId(accessId).build();

        GetAccessKeyLastUsedResponse response = iam.getAccessKeyLastUsed(request);

        System.out.println("Access key was last used at: " +
            response.accessKeyLastUsed().lastUsedDate());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        System.out.println("Done");
    }
```

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

Activer ou désactiver des clés d'accès

Vous pouvez activer ou désactiver une clé d'accès en créant un [UpdateAccessKeyRequest](#) objet, en fournissant l'identifiant de la clé d'accès, éventuellement le nom d'utilisateur et le nom souhaité [status](#), puis en transmettant l'objet de demande à la `IamClient`'s `updateAccessKey` méthode.

Importations

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }
        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

        iam.updateAccessKey(request);

        System.out.printf(
            "Successfully updated the status of access key %s to" +
```

```
        "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Supprimer une clé d'accès

Pour supprimer définitivement une clé d'accès, appelez la `IamClient`'s `deleteKey` méthode en lui fournissant un [DeleteAccessKeyRequest](#) contenant l'identifiant 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 afin qu'elle puisse être réactivée ultérieurement, utilisez plutôt la [updateAccessKey](#) méthode.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {

    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
    }
}
```

```
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

En savoir plus

- [CreateAccessKey](#) dans la référence de IAM l'API
- [ListAccessKeys](#) dans la référence de IAM l'API
- [GetAccessKeyLastUsed](#) dans la référence de IAM l'API
- [UpdateAccessKey](#) dans la référence de IAM l'API
- [DeleteAccessKey](#) dans la référence de IAM l'API

Gérer les IAM utilisateurs

créer un utilisateur ;

Créez un nouvel IAM utilisateur en fournissant le nom d'utilisateur à la `createUser` méthode `IamClient`'s à l'aide d'un [CreateUserRequest](#) objet contenant le nom d'utilisateur.

Importations

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;
```

Code


```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

Répertoire des utilisateurs

Pour répertorier les IAM utilisateurs de votre compte, créez-en un nouveau [ListUsersRequest](#) et passez-le à lamClient la listUsers méthode. Vous pouvez récupérer la liste des utilisateurs en appelant users l'[ListUsersResponse](#) 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 isTruncated de l'objet de réponse. Si celle-ci renvoie true, appelez la méthode marker() de l'objet de réponse. Utilisez la valeur du marqueur de valeur pour créer un nouvel objet demande. Puis, appelez à nouveau la méthode listUsers avec la nouvelle demande.

Importations

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.services.iam.model.User;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void listAllUsers(IamClient iam ) {

    try {

        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListUsersResponse response;

            if (newMarker == null) {
                ListUsersRequest request = ListUsersRequest.builder().build();
                response = iam.listUsers(request);
            } else {
                ListUsersRequest request = ListUsersRequest.builder()
                    .marker(newMarker).build();
                response = iam.listUsers(request);
            }

            for(User user : response.users()) {
                System.out.format("\n Retrieved user %s", user.userName());
            }

            if(!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

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

Mettre à jour un utilisateur

Pour mettre à jour un utilisateur, appelez la `updateUser` méthode de l' `IamClient` objet, qui prend un [UpdateUserRequest](#) objet que vous pouvez utiliser pour modifier le nom ou le chemin de l'utilisateur.

Importations

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

Code

```
public static void updateIAMUser(IamClient iam, String curName, String newName ) {  
  
    try {  
        UpdateUserRequest request = UpdateUserRequest.builder()  
            .userName(curName)  
            .newUserName(newName)  
            .build();  
  
        iam.updateUser(request);  
        System.out.printf("Successfully updated user to username %s",  
            newName);  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

Suppression d'un utilisateur

Pour supprimer un utilisateur, appelez `IamClient` la `deleteUser` demande avec un [UpdateUserRequest](#) objet défini avec le nom d'utilisateur à supprimer.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void deleteIAMUser(IamClient iam, String userName) {

    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("Successfully deleted IAM user " + userName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

En savoir plus

- [IAMUtilisateurs](#) dans le guide de IAM l'utilisateur
- [Gestion des IAM utilisateurs](#) dans le guide de IAM l'utilisateur
- [CreateUser](#) dans la référence de IAM l'API
- [ListUsers](#) dans la référence de IAM l'API
- [UpdateUser](#) dans la référence de IAM l'API
- [DeleteUser](#) dans la référence de IAM l'API

Créez des politiques IAM avec le AWS SDK for Java 2.x

L'[API IAM Policy Builder](#) est une bibliothèque que vous pouvez utiliser pour créer des [politiques IAM](#) en Java et les télécharger vers AWS Identity and Access Management (IAM).

Au lieu de créer une politique IAM en assemblant manuellement une chaîne JSON ou en lisant un fichier, l'API fournit une approche orientée objet côté client pour générer la chaîne JSON. Lorsque vous lisez une politique IAM existante au format JSON, l'API la convertit en [IamPolicy](#) instance à gérer.

L'API IAM Policy Builder est devenue disponible avec la version 2.20.105 du SDK. Utilisez donc cette version ou une version ultérieure dans votre fichier de build Maven. Le numéro de version le plus récent du SDK est [répertorié sur Maven Central](#).

L'extrait suivant montre un exemple de bloc de dépendance pour un fichier pom.xml Maven. Cela vous permet d'utiliser l'API IAM Policy Builder dans votre projet.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>iam-policy-builder</artifactId>
  <version>2.20.139</version>
</dependency>
```

Créer un **IamPolicy**

Cette section présente plusieurs exemples de création de politiques à l'aide de l'API IAM Policy Builder.

Dans chacun des exemples suivants, commencez par les instructions [IamPolicy.Builder](#) et ajoutez une ou plusieurs instructions à l'aide de la `addStatement` méthode. Suivant ce modèle, le [IamStatement.Builder](#) dispose de méthodes pour ajouter l'effet, les actions, les ressources et les conditions à l'instruction.

Exemple : créer une politique basée sur le temps

L'exemple suivant crée une politique basée sur l'identité qui autorise l'action Amazon `GetItem` DynamoDB entre deux points dans le temps.

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.DATE_GREATER_THAN)
                .key("aws:CurrentTime"))
```

```

        .value("2020-04-01T00:00:00Z"))
        .addCondition(b1 -> b1
            .operator(IamConditionOperator.DATE_LESS_THAN)
            .key("aws:CurrentTime")
            .value("2020-06-30T23:59:59Z")))
        .build();

    // Use an IamPolicyWriter to write out the JSON string to a more readable
    format.
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
        .build());
}

```

Sortie JSON

La dernière instruction de l'exemple précédent renvoie la chaîne JSON suivante.

Pour en savoir plus sur cet [exemple](#), consultez le guide de AWS Identity and Access Management l'utilisateur.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:GetItem",
    "Resource" : "*",
    "Condition" : {
      "DateGreaterThan" : {
        "aws:CurrentTime" : "2020-04-01T00:00:00Z"
      },
      "DateLessThan" : {
        "aws:CurrentTime" : "2020-06-30T23:59:59Z"
      }
    }
  }
}

```

Exemple : Spécifier plusieurs conditions

L'exemple suivant montre comment créer une politique basée sur l'identité qui autorise l'accès à des attributs DynamoDB spécifiques. La politique contient deux conditions.

```

public String multipleConditionsExample() {

```

```

    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
            .addAction("dynamodb:Query")
            .addAction("dynamodb:PutItem")
            .addAction("dynamodb:UpdateItem")
            .addAction("dynamodb>DeleteItem")
            .addAction("dynamodb:BatchWriteItem")
            .addResource("arn:aws:dynamodb:*:*:table/table-name")

        .addConditions(IamConditionOperator.STRING_EQUALS.addPrefix("ForAllValues:"),
            "dynamodb:Attributes",
            List.of("column-name1", "column-name2", "column-
name3"))

            .addCondition(b1 ->
b1.operator(IamConditionOperator.STRING_EQUALS.addSuffix("IfExists"))
                .key("dynamodb>Select")
                .value("SPECIFIC_ATTRIBUTES"))

        .build();

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Sortie JSON

La dernière instruction de l'exemple précédent renvoie la chaîne JSON suivante.

Pour en savoir plus sur cet [exemple](#), consultez le guide de AWS Identity and Access Management l'utilisateur.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",
"dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb>DeleteItem",
"dynamodb:BatchWriteItem" ],
    "Resource" : "arn:aws:dynamodb:*:*:table/table-name",
    "Condition" : {
      "ForAllValues:StringEquals" : {

```

```

    "dynamodb:Attributes" : [ "column-name1", "column-name2", "column-name3" ]
  },
  "StringEqualsIfExists" : {
    "dynamodb:Select" : "SPECIFIC_ATTRIBUTES"
  }
}

```

Exemple : Spécifier les principaux

L'exemple suivant montre comment créer une politique basée sur les ressources qui refuse l'accès à un bucket à tous les principaux, à l'exception de ceux spécifiés dans la condition.

```

public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)
            .addResource("arn:aws:s3:::BUCKETNAME/*")
            .addResource("arn:aws:s3:::BUCKETNAME")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.ARN_NOT_EQUALS)
                .key("aws:PrincipalArn")
                .value("arn:aws:iam::444455556666:user/user-name")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Sortie JSON

La dernière instruction de l'exemple précédent renvoie la chaîne JSON suivante.

Pour en savoir plus sur cet [exemple](#), consultez le guide de AWS Identity and Access Management l'utilisateur.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Deny",
    "Principal" : "*",
    "Action" : "s3:*",

```



```

"Resource" : [ "arn:aws:s3:::BUCKETNAME/*", "arn:aws:s3:::BUCKETNAME" ],
"Condition" : {
  "ArnNotEquals" : {
    "aws:PrincipalArn" : "arn:aws:iam::444455556666:user/user-name"
  }
}
}
}
}

```

Exemple : autoriser l'accès entre comptes

L'exemple suivant montre comment autoriser une autre personne Compte AWS à télécharger des objets dans votre compartiment tout en conservant le contrôle total du propriétaire sur les objets chargés.

```

public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS, "111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.STRING_EQUALS)
                .key("s3:x-amz-acl")
                .value("bucket-owner-full-control")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Sortie JSON

La dernière instruction de l'exemple précédent renvoie la chaîne JSON suivante.

Pour en savoir plus sur cet [exemple](#), consultez le guide de l'utilisateur d'Amazon Simple Storage Service.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {

```

```

    "AWS" : "111122223333"
  },
  "Action" : "s3:PutObject",
  "Resource" : "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition" : {
    "StringEquals" : {
      "s3:x-amz-acl" : "bucket-owner-full-control"
    }
  }
}
}
}
}

```

Utiliser et `IamPolicy` avec IAM

Après avoir créé une `IamPolicy` instance, vous utilisez un [IamClient](#) pour travailler avec le service IAM.

L'exemple suivant crée une politique qui permet à une [identité IAM](#) d'écrire des éléments dans une table DynamoDB du compte spécifié avec le paramètre `accountID`. La politique est ensuite téléchargée dans IAM sous forme de chaîne JSON.

```

public String createAndUploadPolicyExample(IamClient iam, String accountID, String
policyName) {
    // Build the policy.
    IamPolicy policy =
        IamPolicy.builder() // 'version' defaults to "2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")
                .addResource("arn:aws:dynamodb:us-east-1:" + accountID
+ ":table/exampleTableName")
                .build())
            .build();
    // Upload the policy.
    iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
    return policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

L'exemple suivant s'appuie sur l'exemple précédent. Le code télécharge la politique et l'utilise comme base pour une nouvelle politique en copiant et en modifiant la déclaration. La nouvelle politique est ensuite téléchargée.

```

public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName, String newPolicyName) {

    String policyArn = "arn:aws:iam::" + accountID + ":policy/" + policyName;
    GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

    String policyVersion = getPolicyResponse.policy().defaultVersionId();
    GetPolicyVersionResponse getPolicyVersionResponse =
        iam.getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

    // Create an IamPolicy instance from the JSON string returned from IAM.
    String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
StandardCharsets.UTF_8);
    IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

    /*
    All IamPolicy components are immutable, so use the copy method that
creates a new instance that
    can be altered in the same method call.

    Add the ability to get an item from DynamoDB as an additional action.
    */
    IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

    // Create a new statement that replaces the original statement.
    IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

    // Upload the new policy. IAM now has both policies.
    iam.createPolicy(r -> r.policyName(newPolicyName)
        .policyDocument(newPolicy.toJson()));

    return newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

IamClient

Les exemples précédents utilisent un `IamClient` argument créé comme indiqué dans l'extrait suivant.

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

Politiques en JSON

Les exemples renvoient les chaînes JSON suivantes.

First example

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:PutItem",
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}
```

Second example

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:PutItem", "dynamodb:GetItem" ],
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}
```

Travailler avec des IAM politiques

Créer une politique

Pour créer une nouvelle politique, indiquez le nom de la stratégie et un document de politique au format JSON dans la méthode a [CreatePolicyRequest](#) to the `IamClient.createPolicy`

Importations

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
```

Code

```
public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

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

Obtenir une politique

Pour récupérer une politique existante, appelez la `getPolicy` méthode `IamClient`'s, en fournissant l'ARN de la politique dans un [GetPolicyRequest](#) objet.

Importations

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void getIAMPolicy(IamClient iam, String policyArn) {

    try {
        GetPolicyRequest request = GetPolicyRequest.builder()
            .policyArn(policyArn).build();

        GetPolicyResponse response = iam.getPolicy(request);
        System.out.format("Successfully retrieved policy %s",
            response.policy().policyName());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Attacher une stratégie de rôle

Vous pouvez associer une politique à un IAM [rôle](#) en appelant la `attachRolePolicy` méthode `IamClient`'s, en lui fournissant le nom du rôle et l'ARN de la politique dans un [AttachRolePolicyRequest](#).

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

Code

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName +
                    " policy is already attached to this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
            AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done");
}
```

```
}
```

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

Répertorier les stratégies de rôle attachées

Répertoriez les politiques associées à un rôle en appelant la `listAttachedRolePolicies` méthode `IamClient`'s. Il faut un [ListAttachedRolePoliciesRequest](#) objet contenant le nom du rôle pour répertorier les politiques.

Appelez `getAttachedPolicies` l'[ListAttachedRolePoliciesResponse](#) objet renvoyé pour obtenir la liste des politiques jointes. Les résultats peuvent être tronqués. Si la méthode `ListAttachedRolePoliciesResponse` de l'objet `isTruncated` renvoie `true`, appelez la méthode `ListAttachedRolePoliciesResponse` de l'objet `marker`. Utilisez le marqueur retourné pour créer une nouvelle demande et utilisez-la pour appeler à nouveau `listAttachedRolePolicies` afin d'obtenir le lot suivant de résultats.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

Code

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
                .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
```



```
List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

// Ensure that the policy is not attached to this role
String polArn = "";
for (AttachedPolicy policy: attachedPolicies) {
    polArn = policy.policyArn();
    if (polArn.compareTo(policyArn)==0) {
        System.out.println(roleName +
            " policy is already attached to this role.");
        return;
    }
}

AttachRolePolicyRequest attachRequest =
    AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

iam.attachRolePolicy(attachRequest);

System.out.println("Successfully attached policy " + policyArn +
    " to role " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done");
}
```

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

Détacher une stratégie de rôle

Pour détacher une politique d'un rôle, appelez la `detachRolePolicy` méthode `IamClient`'s en lui fournissant le nom du rôle et l'ARN de la politique dans un [DetachRolePolicyRequest](#).

Importations

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn )
{
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

En savoir plus

- [Présentation des IAM politiques](#) dans le guide de IAM l'utilisateur.
- [AWS Référence à la politique IAM](#) dans le guide de l'IAM utilisateur.
- [CreatePolicy](#) dans la référence de IAM l'API
- [GetPolicy](#) dans la référence de IAM l'API
- [AttachRolePolicy](#) dans la référence de IAM l'API
- [ListAttachedRolePolicies](#) dans la référence de IAM l'API
- [DetachRolePolicy](#) dans la référence de IAM l'API

Travailler avec des certificats IAM de serveur

Pour activer les connexions HTTPS à votre site Web ou à votre application AWS, vous avez besoin d'un certificat de serveur SSL/TLS. Vous pouvez utiliser un certificat de serveur fourni par un fournisseur externe AWS Certificate Manager ou un certificat que vous avez obtenu auprès d'un fournisseur externe.

Nous vous recommandons de les utiliser ACM pour provisionner, gérer et déployer vos certificats de serveur. ACM Vous pouvez ainsi demander un certificat, le déployer sur vos AWS ressources et nous laisser ACM gérer le renouvellement des certificats pour vous. Les certificats fournis par ACM sont gratuits. Pour plus d'informations ACM, consultez le [guide de AWS Certificate Manager l'utilisateur](#).

Obtenir un certificat de serveur

Vous pouvez récupérer un certificat de serveur en appelant la `getServerCertificate` méthode `IamClient`'s et en [GetServerCertificateRequest](#) lui transmettant le nom du certificat.

Importations

```
import software.amazon.awssdk.services.iam.model.GetServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.GetServerCertificateResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void getCertificate(IamClient iam,String certName ) {

    try {
        GetServerCertificateRequest request = GetServerCertificateRequest.builder()
            .serverCertificateName(certName)
            .build();

        GetServerCertificateResponse response = iam.getServerCertificate(request);
        System.out.format("Successfully retrieved certificate with body %s",
            response.serverCertificate().certificateBody());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

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

Lister les certificats de serveur

Pour répertorier les certificats de votre serveur, appelez la `listServerCertificates` méthode `IamClient`'s avec un [ListServerCertificatesRequest](#). Elle renvoie un [ListServerCertificatesResponse](#).

Appelez la `serverCertificateMetadataList` méthode de `ListServerCertificateResponse` l'objet renvoyé pour obtenir une liste d'[ServerCertificateMetadata](#) objets que vous pouvez utiliser pour obtenir des informations sur chaque certificat.

Les résultats peuvent être tronqués. Si la méthode `ListServerCertificateResponse` de l'objet `isTruncated` renvoie `true`, appelez la méthode `ListServerCertificatesResponse` de l'objet `marker` et utilisez le marqueur pour créer une nouvelle demande. Utilisez la nouvelle demande pour appeler à nouveau `listServerCertificates` et obtenir le lot suivant de résultats.

Importations

```
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.ListServerCertificatesRequest;  
import software.amazon.awssdk.services.iam.model.ListServerCertificatesResponse;  
import software.amazon.awssdk.services.iam.model.ServerCertificateMetadata;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
public static void listCertificates(IamClient iam) {  
  
    try {  
        boolean done = false;  
        String newMarker = null;  
  
        while(!done) {  
            ListServerCertificatesResponse response;  
  
            if (newMarker == null) {  
                ListServerCertificatesRequest request =
```

```
        ListServerCertificatesRequest.builder().build());
        response = iam.listServerCertificates(request);
    } else {
        ListServerCertificatesRequest request =
            ListServerCertificatesRequest.builder()
                .marker(newMarker).build();
        response = iam.listServerCertificates(request);
    }

    for(ServerCertificateMetadata metadata :
        response.serverCertificateMetadataList()) {
        System.out.printf("Retrieved server certificate %s",
            metadata.serverCertificateName());
    }

    if(!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

Mettre à jour un certificat de serveur

Vous pouvez mettre à jour le nom ou le chemin d'un certificat de serveur en appelant la `updateServerCertificate` méthode `IamClient`'s. Il faut utiliser un ensemble d'[UpdateServerCertificateRequest](#) objets portant le nom actuel du certificat de serveur et un nouveau nom ou un nouveau chemin.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateRequest;
```

```
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateResponse;
```

Code

```
public static void updateCertificate(IamClient iam, String curName, String newName)
{
    try {
        UpdateServerCertificateRequest request =
            UpdateServerCertificateRequest.builder()
                .serverCertificateName(curName)
                .newServerCertificateName(newName)
                .build();

        UpdateServerCertificateResponse response =
            iam.updateServerCertificate(request);

        System.out.printf("Successfully updated server certificate to name %s",
            newName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Supprimer un certificat de serveur

Pour supprimer un certificat de serveur, appelez la `deleteServerCertificate` méthode `IamClient`'s avec un [DeleteServerCertificateRequest](#) contenant le nom du certificat.

Imports

```
import software.amazon.awssdk.services.iam.model.DeleteServerCertificateRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
public static void deleteCert(IamClient iam,String certName ) {

    try {
        DeleteServerCertificateRequest request =
            DeleteServerCertificateRequest.builder()
                .serverCertificateName(certName)
                .build();

        iam.deleteServerCertificate(request);
        System.out.println("Successfully deleted server certificate " +
            certName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

En savoir plus

- [Utilisation des certificats de serveur](#) dans le guide de IAM l'utilisateur
- [GetServerCertificate](#) dans la référence de IAM l'API
- [ListServerCertificates](#) dans la référence de IAM l'API
- [UpdateServerCertificate](#) dans la référence de IAM l'API
- [DeleteServerCertificate](#) dans la référence de IAM l'API
- [AWS Certificate Manager Guide de l'utilisateur](#)

Travaillez avec Kinesis

Cette section fournit des exemples de programmation à l'[Amazon Kinesis](#) aide de la version AWS SDK for Java 2.x.

Pour plus d'informations Kinesis, consultez le [guide du Amazon Kinesis développeur](#).

Les exemples suivants 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

- [S'abonner à Amazon Kinesis Data Streams](#)

S'abonner à Amazon Kinesis Data Streams

Les exemples suivants vous montrent comment récupérer et traiter des données à partir de flux de Amazon Kinesis données à l'aide de `subscribeToShard` cette méthode. Kinesis Data Streams utilise désormais la fonction fanout améliorée et une API de récupération de données HTTP/2 à faible latence, ce qui permet aux développeurs d'exécuter plus facilement plusieurs applications à faible latence et hautes performances sur le même flux de données. Kinesis

Configuration

Créez d'abord un Kinesis client asynchrone et un [SubscribeToShardRequest](#) objet. Ces objets sont utilisés dans chacun des exemples suivants pour s'abonner à des événements Kinesis.

Importations

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicInteger;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEventStream;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponse;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

Code

```
Region region = Region.US_EAST_1;
KinesisAsyncClient client = KinesisAsyncClient.builder()
    .region(region)
    .build();
```



```
SubscribeToShardRequest request = SubscribeToShardRequest.builder()
    .consumerARN(CONSUMER_ARN)
    .shardId("arn:aws:kinesis:us-east-1:111122223333:stream/
StockTradeStream")
    .startingPosition(s -> s.type(ShardIteratorType.LATEST)).build();
```

Utiliser l'interface du générateur

Vous pouvez utiliser `builder` cette méthode pour simplifier la création du [SubscribeToShardResponseHandler](#).

À l'aide du générateur, vous pouvez définir chaque rappel de cycle de vie avec un appel de méthode au lieu d'implémenter l'intégralité de l'interface.

Code

```
private static CompletableFuture<Void> responseHandlerBuilder(KinesisAsyncClient
client, SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onComplete(() -> System.out.println("All records stream
successfully"))
    // Must supply some type of subscriber
    .subscriber(e -> System.out.println("Received event - " + e))
    .build();
    return client.subscribeToShard(request, responseHandler);
}
```

Pour plus de contrôle sur l'éditeur, vous pouvez utiliser la méthode `publisherTransformer` pour personnaliser l'éditeur.

Code

```
private static CompletableFuture<Void>
responseHandlerBuilderPublisherTransformer(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
```

```

        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .publisherTransformer(p -> p.filter(e -> e instanceof
SubscribeToShardEvent).limit(100))
        .subscriber(e -> System.out.println("Received event - " + e))
        .build();
    return client.subscribeToShard(request, responseHandler);
}

```

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

Utiliser un gestionnaire de réponses personnalisé

Pour un contrôle total de l'abonné et de l'éditeur, implémentez l'`SubscribeToShardResponseHandler` interface.

Dans cet exemple, vous implémentez la méthode `onEventStream` qui autorise un accès complet à l'éditeur. Cet exemple montre comment transformer l'éditeur en enregistrements d'événement pour impression par l'abonné.

Code

```

    private static CompletableFuture<Void>
responseHandlerBuilderClassic(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {

        @Override
        public void responseReceived(SubscribeToShardResponse response) {
            System.out.println("Receieved initial response");
        }

        @Override
        public void onEventStream(SdkPublisher<SubscribeToShardEventStream>
publisher) {
            publisher
                // Filter to only SubscribeToShardEvents
                .filter(SubscribeToShardEvent.class)
                // Flat map into a publisher of just records
                .flatMapIterable(SubscribeToShardEvent::records)
                // Limit to 1000 total records
                .limit(1000)

```

```

        // Batch records into lists of 25
        .buffer(25)
        // Print out each record batch
        .subscribe(batch -> System.out.println("Record Batch - " +
batch));
    }

    @Override
    public void complete() {
        System.out.println("All records stream successfully");
    }

    @Override
    public void exceptionOccurred(Throwable throwable) {
        System.err.println("Error during stream - " + throwable.getMessage());
    }
};
return client.subscribeToShard(request, responseHandler);
}

```

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

Utiliser l'interface visiteur

Vous pouvez utiliser un objet [Visitor](#) pour vous abonner à des événements spécifiques que vous souhaitez surveiller.

Code

```

private static CompletableFuture<Void>
responseHandlerBuilderVisitorBuilder(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler.Visitor visitor =
SubscribeToShardResponseHandler.Visitor
        .builder()
        .onSubscribeToShardEvent(e -> System.out.println("Received subscribe to
shard event " + e))
        .build();
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))

```

```
        .subscriber(visitor)
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

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

Utiliser un abonné personnalisé

Vous pouvez également implémenter votre propre abonné personnalisé pour s'abonner au flux.

Cet extrait de code montre un exemple d'abonné.

Code

```
private static class MySubscriber implements
Subscriber<SubscribeToShardEventStream> {

    private Subscription subscription;
    private AtomicInteger eventCount = new AtomicInteger(0);

    @Override
    public void onSubscribe(Subscription subscription) {
        this.subscription = subscription;
        this.subscription.request(1);
    }

    @Override
    public void onNext(SubscribeToShardEventStream shardSubscriptionEventStream) {
        System.out.println("Received event " + shardSubscriptionEventStream);
        if (eventCount.incrementAndGet() >= 100) {
            // You can cancel the subscription at any time if you wish to stop
receiving events.
            subscription.cancel();
        }
        subscription.request(1);
    }

    @Override
    public void onError(Throwable throwable) {
        System.err.println("Error occurred while stream - " +
throwable.getMessage());
    }
}
```

```
@Override
public void onComplete() {
    System.out.println("Finished streaming all events");
}
}
```

Vous pouvez transmettre l'abonné personnalisé à la `subscribe` méthode comme indiqué dans l'extrait de code suivant.

Code

```
private static CompletableFuture<Void>
responseHandlerBuilderSubscriber(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(MySubscriber::new)
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

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

Écrire des enregistrements de données dans un flux Kinesis de données

Vous pouvez utiliser l'[KinesisClient](#) objet pour écrire des enregistrements de données dans un flux de Kinesis données à l'aide de la `putRecords` méthode. Pour appeler correctement cette méthode, créez un [PutRecordsRequest](#) objet. Vous transmettez le nom du flux de données à la `streamName` méthode. Vous devez également transmettre les données en utilisant la méthode `putRecords` (comme indiqué dans l'exemple de code suivant).

Importations

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
```

```
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
```

Dans l'exemple de code Java suivant, notez que `StockTrade` l'objet est utilisé comme donnée pour écrire dans le flux de Kinesis données. Avant d'exécuter cet exemple, assurez-vous d'avoir créé le flux de données.

Code

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to which records are
                written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
```

```
KinesisClient kinesisClient = KinesisClient.builder()
    .region(region)
    .build();

// Ensure that the Kinesis Stream is valid.
validateStream(kinesisClient, streamName);
setStockData(kinesisClient, streamName);
kinesisClient.close();
}

public static void setStockData(KinesisClient kinesisClient, String streamName) {
    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x = 0; x < index; x++) {
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
    String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization by
    // the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
```

```

        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as
the partition key, explained in
                                                // the Supplemental Information
section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        System.err.println(e.getMessage());
    }
}

private static void validateStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }

    } catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
        System.err.println(e);
        System.exit(1);
    }
}
}

```


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

Utiliser une bibliothèque tierce

Vous pouvez utiliser d'autres bibliothèques tierces au lieu d'implémenter un abonné personnalisé. Cet exemple montre comment utiliser l' RxJava implémentation, mais vous pouvez utiliser n'importe quelle bibliothèque qui implémente les interfaces Reactive Streams. Consultez la [page RxJava wiki sur Github](#) pour plus d'informations sur cette bibliothèque.

Pour utiliser la bibliothèque, ajoutez-la en tant que dépendance. Si vous utilisez Maven, l'exemple illustre l'extrait POM à utiliser.

Entrée POM

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>2.1.14</version>
</dependency>
```

Importations

```
import java.net.URI;
import java.util.concurrent.CompletableFuture;

import io.reactivex.Flowable;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.http.Protocol;
import software.amazon.awssdk.http.SdkHttpConfigurationOption;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.StartingPosition;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
import software.amazon.awssdk.utils.AttributeMap;
```

Cet exemple est utilisé RxJava dans la méthode du `onEventStream` cycle de vie. Cet exemple vous donne un accès complet à l'éditeur, qui peut être utilisé pour créer un Rx Flowable.

Code

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onEventStream(p -> Flowable.fromPublisher(p)
        .ofType(SubscribeToShardEvent.class)

.flatMapIterable(SubscribeToShardEvent::records)
        .limit(1000)
        .buffer(25)
        .subscribe(e -> System.out.println("Record
batch = " + e)))
    .build();
```

Vous pouvez également utiliser la méthode `publisherTransformer` avec l'éditeur `Flowable`. Vous devez adapter l'éditeur `Flowable` à un `SdkPublisher`, comme indiqué dans l'exemple suivant.

Code

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
    .build();
```

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

En savoir plus

- [SubscribeToShardEvent](#) dans la référence de Amazon Kinesis l'API
- [SubscribeToShard](#) dans la référence de Amazon Kinesis l'API

AWS Lambda Fonctions d'appel, de liste et de suppression

Cette section fournit des exemples de programmation avec le client de Lambda service à l'aide de la version AWS SDK for Java 2.x.

Rubriques

- [Appel d'une fonction Lambda](#)
- [Liste des fonctions Lambda](#)
- [Suppression d'une fonction Lambda](#)

Appel d'une fonction Lambda

Vous pouvez invoquer une Lambda fonction en créant un [LambdaClient](#) objet et en invoquant sa `invoke` méthode. Créez un [InvokeRequest](#) objet pour spécifier des informations supplémentaires telles que le nom de la fonction et la charge utile à transmettre à la Lambda fonction. Les noms des fonctions apparaissent sous la forme `arn:aws:lambda:us-east-1:123456789012:function::HelloFunction` Vous pouvez récupérer la valeur en consultant la fonction dans le AWS Management Console.

Pour transmettre des données de charge utile à une fonction, créez un [SdkBytes](#) objet contenant des informations. Par exemple, dans l'exemple de code suivant, notez les données JSON transmises à la fonction Lambda .

Importations

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Code

L'exemple de code suivant montre comment invoquer une Lambda fonction.

```
public static void invokeFunction(LambdaClient awsLambda, String functionName) {
```

```
    InvokeResponse res = null ;
    try {
        //Need a SdkBytes instance for the payload
        String json = "{\"Hello \":\"Paris\"}";
        SdkBytes payload = SdkBytes.fromUtf8String(json) ;

        //Setup an InvokeRequest
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String() ;
        System.out.println(value);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Liste des fonctions Lambda

Créez un [Lambda Client](#) objet et invoquez sa `listFunctions` méthode. Cette méthode renvoie un [ListFunctionsResponse](#) objet. Vous pouvez invoquer la `functions` méthode de cet objet pour renvoyer une liste d'[FunctionConfiguration](#) objets. 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 software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;
```

Code

L'exemple de code Java suivant illustre comment récupérer une liste de noms de fonctions.

```
public static void listFunctions(LambdaClient awsLambda) {

    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();

        for (FunctionConfiguration config: list) {
            System.out.println("The function name is "+config.functionName());
        }

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

Suppression d'une fonction Lambda

Créez un [LambdaClient](#) objet et invoquez sa `deleteFunction` méthode. Créez un [DeleteFunctionRequest](#) objet et transmettez-le à la `deleteFunction` méthode. Cet objet contient des informations telles que le nom de la fonction à supprimer. Les noms des fonctions apparaissent sous la forme `arn:aws:lambda:us-east-1:123456789012:function::HelloFunction` Vous pouvez récupérer la valeur en consultant la fonction dans le AWS Management Console.

Importations

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Code

Le code Java suivant montre comment supprimer une Lambda fonction.

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName ) {
```

```
try {
    DeleteFunctionRequest request = DeleteFunctionRequest.builder()
        .functionName(functionName)
        .build();

    awsLambda.deleteFunction(request);
    System.out.println("The "+functionName+" function was deleted");

} catch(LambdaException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

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

Travaillez avec Amazon S3

Cette section fournit des exemples de programmation avec [Amazon Simple Storage Service \(S3\)](#) à l'aide du AWS SDK for Java 2.x

Les exemples suivants 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.

Note

À partir de la version 2.18.x, l'adressage de type hôte virtuel est AWS SDK for Java 2.x utilisé pour inclure un remplacement du point de terminaison. Cela s'applique tant que le nom du bucket est une étiquette DNS valide.

Appelez la [forcePathStyle](#) méthode `true` dans votre générateur de clients pour forcer le client à utiliser un adressage de type chemin pour les buckets.

L'exemple suivant montre un client de service configuré avec un remplacement de point de terminaison et utilisant un adressage de type chemin.

```
S3Client client = S3Client.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create("https://s3.us-
west-2.amazonaws.com"))
```

```
.forcePathStyle(true)
.build();
```

Utiliser des points d'accès ou des points d'accès multirégionaux

Une fois les [points d'accès Amazon S3](#) ou les [points d'accès multirégionaux](#) configurés, vous pouvez appeler des méthodes d'objet, telles que `putObject` et `getObject` et fournir l'identifiant du point d'accès au lieu d'un nom de compartiment.

Par exemple, si l'identifiant ARN d'un point d'accès est `arn:aws:s3:us-west-2:123456789012:accesspoint/test`, vous pouvez utiliser l'extrait de code suivant pour appeler la `putObject` méthode.

```
Path path = Paths.get(URI.create("file:///temp/file.txt"));

s3Client.putObject(builder -> builder
    .key("myKey")
    .bucket("arn:aws:s3:us-west-2:123456789012:accesspoint/test")
    , path);
```

À la place de la chaîne ARN, vous pouvez également utiliser l'[alias de type bucket](#) du point d'accès pour le bucket paramètre.

Pour utiliser un point d'accès multirégional, remplacez le bucket paramètre par l'ARN du point d'accès multirégional au format suivant.

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

Ajoutez la dépendance Maven suivante pour utiliser les points d'accès multirégionaux à l'aide du SDK for Java. Recherchez la [dernière version](#) sur Maven Central.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>auth-crt</artifactId>
  <version>VERSION</version>
</dependency>
```

Rubriques

- [Création, liste et suppression de Amazon S3 buckets](#)
- [Travailler avec des Amazon S3 objets](#)
- [Travailler avec des URL Amazon S3 pré-signées](#)
- [Accès entre régions pour Amazon S3](#)
- [Sommes de contrôle Amazon S3 avec la 3](#)
- [Utiliser un client S3 performant : client S3 AWS basé sur CRT](#)
- [Transférez des fichiers et des répertoires avec Amazon S3 Transfer Manager](#)

Création, liste et suppression de Amazon S3 buckets

Chaque objet (fichier) dans Amazon S3 doit résider dans un compartiment. Un compartiment représente un ensemble (conteneur) d'objets. Chaque compartiment doit avoir une clé (nom) unique. Pour obtenir des informations détaillées sur les buckets et leur configuration, consultez la section [Utilisation des Amazon S3 buckets](#) dans le Guide de l'Amazon Simple Storage Service utilisateur.

Note

Bonne pratique

Nous vous recommandons d'activer la règle du [AbortIncompleteMultipartUpload](#) cycle de vie sur vos Amazon S3 buckets.

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 plus d'informations, consultez la section [Configuration du cycle de vie d'un bucket avec gestion des versions](#) dans le guide de l'Amazon Simple Storage Service utilisateur.

Note

Ces extraits de code supposent que vous comprenez le contenu de base et que vous avez configuré les informations d'AWS identification par défaut à l'aide des informations contenues dans [the section called "Configuration de l'accès par authentification unique pour le SDK"](#)

Création d'un compartiment

Créez un [CreateBucketRequest](#) et fournissez un nom de compartiment. Passez-le à la méthode du `S3Client`. `createBucket` Utilisez le client `S3Client` pour effectuer d'autres opérations telles que l'affichage ou la suppression de compartiments, comme illustré ultérieurement dans les exemples.

Importations

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

Créez d'abord un client S3.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Effectuez une demande de création de compartiment.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketOps {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String bucket = "bucket" + System.currentTimeMillis();
        System.out.println(bucket);
        createBucket(s3, bucket);
        performOperations(s3, bucket);
    }

    // Create a bucket by using a S3Waiter object
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
    }  
}
```

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

Lister les compartiments

Construisez un [ListBucketsRequest](#). Utilisez la `listBuckets` méthode du `S3Client` pour récupérer la liste des buckets. Si la demande aboutit, un [ListBucketsResponse](#) est renvoyé. Utilisez cet objet réponse pour récupérer la liste des compartiments.

Importations

```
import software.amazon.awssdk.core.waiters.WaiterResponse;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;  
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;  
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;  
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;  
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;  
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

Créez d'abord un client S3.

```
Region region = Region.US_EAST_1;  
S3Client s3 = S3Client.builder()  
    .region(region)  
    .build();
```

Effectuez une demande de création de la liste des compartiments.

```
// List buckets  
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();  
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);  
listBucketsResponse.buckets().stream().forEach(x ->  
System.out.println(x.name()));
```

Consultez 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 le service retourne une erreur. S'il s'agit d'un [compartiment avec gestion des versions](#), vous devez également supprimer tous les objets versionnés qui y figurent.

Rubriques

- [Supprimer des objets dans un compartiment](#)
- [Supprimer un compartiment vide](#)

Supprimer des objets dans un compartiment

Créez une [ListObjectsV2Request](#) et utilisez la `listObjects` méthode du `S3Client` pour récupérer la liste des objets du bucket. Ensuite, utilisez la méthode `deleteObject` sur chaque objet pour le supprimer.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

Code

Créez d'abord un client S3.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Supprimez tous les objets du compartiment.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketDeletion {
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <bucket>

            Where:
                bucket - The bucket to delete (for example, bucket1).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteObjectsInBucket(s3, bucket);
        s3.close();
    }
}
```

```
public static void deleteObjectsInBucket(S3Client s3, String bucket) {
    try {
        // To delete a bucket, all the objects in the bucket must be deleted first.
        ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
            .bucket(bucket)
            .build();
        ListObjectsV2Response listObjectsV2Response;

        do {
            listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
            for (S3Object s3Object : listObjectsV2Response.contents()) {
                DeleteObjectRequest request = DeleteObjectRequest.builder()
                    .bucket(bucket)
                    .key(s3Object.key())
                    .build();
                s3.deleteObject(request);
            }
        } while (listObjectsV2Response.isTruncated());
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder().bucket(bucket).build();
        s3.deleteBucket(deleteBucketRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

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

Supprimer un compartiment vide

Créez un [DeleteBucketRequest](#) avec un nom de bucket et transmettez-le à la méthode du S3Client. `deleteBucket`

Importations

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

Créez d'abord un client S3.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Supprimez le compartiment.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

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

Travailler avec des Amazon S3 objets

Un objet Amazon S3 représente un fichier ou un ensemble de données. Chaque objet doit être contenu dans un [compartiment](#).

Note

Bonne pratique

Nous vous recommandons d'activer la règle du [AbortIncompleteMultipartUpload](#) cycle de vie sur vos Amazon S3 buckets.

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 plus d'informations, consultez la section [Configuration du cycle de vie d'un bucket avec gestion des versions](#) dans le guide de l'Amazon Simple Storage Service utilisateur.

Note

Ces extraits de code supposent que vous comprenez le contenu de base et que vous avez configuré les informations d'AWS identification par défaut à l'aide des informations contenues dans [the section called "Configuration de l'accès par authentification unique pour le SDK"](#)

Rubriques

- [Charger un objet](#)
- [Charger des objets partitionnés](#)
- [Supprimer un objet](#)
- [Répertorier les objets](#)
- [Plus d'exemples](#)

Charger un objet

Créez un [PutObjectRequest](#) et fournissez un nom de compartiment et un nom de clé. Utilisez ensuite la `putObject` méthode du `S3Client` avec un [RequestBody](#) qui contient le contenu de l'objet et l'`PutObjectRequest` objet. Le compartiment doit exister, sans quoi le service renvoie une erreur.

Importations

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
```



```
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
Region region = Region.US_WEST_2;
s3 = S3Client.builder()
    .region(region)
    .build();

createBucket(s3, bucketName, region);

PutObjectRequest objectRequest = PutObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.putObject(objectRequest,
    RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

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

Charger des objets partitionnés

Utilisez la `createMultipartUpload` méthode du `S3Client` pour obtenir un identifiant de téléchargement. Ensuite, utilisez la méthode `uploadPart` pour charger chaque partie. Enfin, utilisez

la `completeMultipartUpload` méthode du `S3Client` pour demander de fusionner toutes les parties téléchargées et de terminer l'opération de téléchargement. Amazon S3

Importations

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
        // First create a multipart upload and get the upload id
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```

```
        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(1).build();

        String etag1 = s3
                .uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
                .eTag();

        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
                .uploadId(uploadId)
                .partNumber(2).build();

        String etag2 = s3
                .uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
                .eTag();

        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Finally call completeMultipartUpload operation to tell S3 to merge
all
        // uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
```

```
.uploadId(uploadId)
.multipartUpload(completedMultipartUpload)
.build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
```

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

Supprimer un objet

Créez un [DeleteObjectRequest](#) et fournissez un nom de compartiment et un nom de clé. Utilisez la `deleteObject` méthode du `S3Client` et transmettez-lui le nom du bucket et de l'objet à supprimer. Le compartiment et la clé d'objet spécifiés doivent exister, sans quoi le service renvoie une erreur.

Importations

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.deleteObject(deleteObjectRequest);
```

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

Copier un objet

Créez un [CopyObjectRequest](#) et fournissez un nom de compartiment dans lequel l'objet est copié, une valeur de chaîne codée en URL (voir la méthode `URLEncoder.encode`) et le nom clé de l'objet. Utilisez la `copyObject` méthode du `S3Client` et transmettez l'[CopyObjectRequest](#) objet. Le compartiment et la clé d'objet spécifiés doivent exister, sans quoi le service renvoie une erreur.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
bucket2).

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        try {
```

```
CopyObjectResponse copyRes = s3.copyObject(copyReq);
return copyRes.copyObjectResult().toString();

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
}
```

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

Répertorier les objets

Créez un [ListObjectsRequest](#) et fournissez le nom du compartiment. Invoquez ensuite la `listObjects` méthode du `S3Client` et transmettez l'`ListObjectsRequest` objet. Cette méthode renvoie un [ListObjectsResponse](#) contenant tous les objets du compartiment. Vous pouvez appeler la méthode `contents` de cet objet pour obtenir une liste d'objets. Vous pouvez parcourir cette liste pour afficher les objets, comme indiqué dans l'exemple de code suivant.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;
```

Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
```

```
public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                .builder()
                .bucket(bucketName)
                .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
```



```
        System.out.print("\n The name of the key is " + myValue.key());
        System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
        System.out.print("\n The owner is " + myValue.owner());
    }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calKb(Long val) {
    return val / 1024;
}
}
```

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

Plus d'exemples

La section [Exemples de code](#) de ce guide contient d'autres exemples d'utilisation des objets Amazon S3, notamment comment [télécharger un objet](#).

Travailler avec des URL Amazon S3 pré-signées

Les URL pré-signées fournissent un accès temporaire aux objets S3 privés sans que les utilisateurs aient besoin AWS d'informations d'identification ou d'autorisations.

Par exemple, supposons qu'Alice ait accès à un objet S3 et qu'elle souhaite partager temporairement l'accès à cet objet avec Bob. Alice peut générer une requête GET pré-signée à partager avec Bob afin qu'il puisse télécharger l'objet sans avoir besoin d'accéder aux informations d'identification d'Alice. Vous pouvez générer des URL pré-signées pour les requêtes HTTP GET et HTTP PUT.

Générez une URL pré-signée pour un objet, puis téléchargez-le (requête GET)

L'exemple suivant comprend deux parties.

- Partie 1 : Alice génère l'URL pré-signée d'un objet.
- Partie 2 : Bob télécharge l'objet à l'aide de l'URL pré-signée.

Partie 1 : Génération de l'URL

Alice possède déjà un objet dans un compartiment S3. Elle utilise le code suivant pour générer une chaîne d'URL que Bob pourra utiliser dans une requête GET ultérieure.

Importations

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
```

```

        .key(keyName)
        .build();

    GetObjectPresignRequest presignRequest = GetObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(10)) // The URL will expire
in 10 minutes.
        .getObjectRequest(objectRequest)
        .build();

    PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
    logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
    logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

    return presignedRequest.url().toExternalForm();
}
}

```

Partie 2 : Télécharger l'objet

Bob utilise l'une des trois options de code suivantes pour télécharger l'objet. Il pourrait également utiliser un navigateur pour exécuter la requête GET.

Utiliser le JDK **URLConnection** (depuis la v1.1)

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

```

    }
    return byteArrayOutputStream.toByteArray();
}

```

Utiliser le JDK `HttpClient` (depuis la v11)

```

/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());
    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Utilisation `SdkHttpClient` depuis le SDK pour Java

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())

```

```
        .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
                sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
                            byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
                response.httpResponse().statusCode());
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
}
```

Consultez l'[exemple complet](#) et [testez](#) GitHub.

Générez une URL pré-signée pour un téléchargement, puis téléchargez un fichier (requête PUT)

L'exemple suivant comprend deux parties.

- Partie 1 : Alice génère l'URL pré-signée pour télécharger un objet.
- Partie 2 : Bob télécharge un fichier à l'aide de l'URL pré-signée.

Partie 1 : Génération de l'URL

Alice possède déjà un compartiment S3. Elle utilise le code suivant pour générer une chaîne d'URL que Bob pourra utiliser dans une requête PUT ultérieure.

Importations

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

```
/* Create a presigned URL to use in a subsequent PUT request */
```

```

public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires in
10 minutes.
            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Partie 2 : Chargement d'un objet de fichier

Bob utilise l'une des trois options de code suivantes pour charger un fichier.

Utiliser le JDK **HttpURLConnection** (depuis la v1.1)

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useHttpURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" + k,
v));
    }
}

```

```

connection.setRequestMethod("PUT");
OutputStream out = connection.getOutputStream();

try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
    FileChannel inChannel = file.getChannel()) {
    ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

    while (inChannel.read(buffer) > 0) {
        buffer.flip();
        for (int i = 0; i < buffer.limit(); i++) {
            out.write(buffer.get());
        }
        buffer.clear();
    }
} catch (IOException e) {
    logger.error(e.getMessage(), e);
}

out.close();
connection.getResponseCode();
logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Utiliser le JDK `HttpClient` (depuis la v11)

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())
            .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))))
    }
}

```



```

        .build(),
        HttpResponse.BodyHandlers.discarding());

    logger.info("HTTP response code is " + response.statusCode());
} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Utilisation **SdkHttpClient** depuis le SDK pour Java

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k, v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}", response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
}

```

Consultez l'[exemple complet](#) et [testez](#) GitHub.

Accès entre régions pour Amazon S3

Lorsque vous travaillez avec des compartiments Amazon Simple Storage Service (Amazon S3), vous connaissez généralement Région AWS le compartiment correspondant. La région avec laquelle vous travaillez est déterminée lorsque vous créez le client S3.

Cependant, il peut arriver que vous deviez travailler avec un compartiment spécifique, mais vous ne savez pas s'il se trouve dans la même région que celle définie pour le client S3.

Au lieu de passer d'autres appels pour déterminer la région du compartiment, vous pouvez utiliser le SDK pour permettre l'accès aux compartiments S3 dans différentes régions.

Installation

Support pour l'accès interrégional est devenu disponible avec la version 2.20.111 du SDK. Utilisez cette version ou une version ultérieure dans votre fichier de build Maven pour la s3 dépendance, comme indiqué dans l'extrait suivant.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.111</version>
</dependency>
```

Ensuite, lorsque vous créez votre client S3, activez l'accès entre régions comme indiqué dans l'extrait de code. Par défaut, l'accès n'est pas activé.

```
S3AsyncClient client = S3AsyncClient.builder()
    .crossRegionAccessEnabled(true)
    .build();
```

Comment le SDK fournit un accès interrégional

Lorsque vous faites référence à un compartiment existant dans une demande, par exemple lorsque vous utilisez la `putObject` méthode, le SDK lance une demande auprès de la région configurée pour le client.

Si le compartiment n'existe pas dans cette région spécifique, la réponse d'erreur inclut la région dans laquelle réside le compartiment. Le SDK utilise ensuite la région appropriée dans une deuxième demande.

Pour optimiser les futures demandes adressées au même compartiment, le SDK met en cache ce mappage de région dans le client.

Considérations

Lorsque vous activez l'accès au compartiment entre régions, sachez que le premier appel d'API peut entraîner une augmentation de la latence si le compartiment ne se trouve pas dans la région configurée par le client. Cependant, les appels suivants bénéficient des informations de région mises en cache, ce qui améliore les performances.

Lorsque vous activez l'accès entre régions, l'accès au compartiment n'est pas affecté. L'utilisateur doit être autorisé à accéder au bucket, quelle que soit la région dans laquelle il réside.

Sommes de contrôle Amazon S3 avec la 3

Amazon Simple Storage Service (Amazon S3) permet de spécifier une somme de contrôle lorsque vous chargez un objet. Lorsque vous spécifiez une somme de contrôle, elle est stockée avec l'objet et peut être validée lors du téléchargement de l'objet.

Les checksums fournissent une couche supplémentaire d'intégrité des données lorsque vous transférez des fichiers. Avec les checksums, vous pouvez vérifier la cohérence des données en confirmant que le fichier reçu correspond au fichier d'origine. Pour plus d'informations sur les sommes de contrôle avec Amazon S3, consultez le [guide de l'utilisateur d'Amazon Simple Storage Service](#).

Amazon S3 prend actuellement en charge quatre algorithmes de somme de contrôle : SHA-1, SHA-256, CRC-32 et CRC-32C. Vous avez la possibilité de choisir l'algorithme qui répond le mieux à vos besoins et de laisser le SDK calculer le checksum. Vous pouvez également spécifier leur propre valeur de somme de contrôle précalculée à l'aide de l'un des quatre algorithmes pris en charge.

Nous discutons des sommes de contrôle en deux phases de demande : le téléchargement d'un objet et le téléchargement d'un objet.

Charger un objet

Les valeurs valides pour l'algorithme sont CRC32CRC32C,SHA1, etSHA256.

L'extrait de code suivant montre une demande de téléchargement d'un objet avec une somme de contrôle CRC-32. Lorsque le SDK envoie la demande, il calcule le checksum CRC-32 et télécharge l'objet. Amazon S3 stocke le checksum avec l'objet.

Si la somme de contrôle calculée par le SDK ne correspond pas à la somme de contrôle calculée par Amazon S3 lorsqu'il reçoit la demande, une erreur est renvoyée.

Utiliser une valeur de somme de contrôle précalculée

Une valeur de somme de contrôle précalculée fournie avec la demande désactive le calcul automatique par le SDK et utilise la valeur fournie à la place.

L'exemple suivant montre une demande avec une somme de contrôle SHA-256 précalculée.

Si Amazon S3 détermine que la valeur de la somme de contrôle est incorrecte pour l'algorithme spécifié, le service renvoie une réponse d'erreur.

Chargements partitionnés

Vous pouvez également utiliser des checksums pour les téléchargements partitionnés.

Télécharger un objet

Lorsque vous utilisez la méthode [GetObject](#) pour télécharger un objet, le SDK valide automatiquement le checksum est. `enabled`

La demande contenue dans l'extrait suivant demande au SDK de valider la somme de contrôle dans la réponse en calculant la somme de contrôle et en comparant les valeurs.

Si l'objet n'a pas été chargé avec une somme de contrôle, aucune validation n'a lieu.

Un objet dans Amazon S3 peut avoir plusieurs checksum, mais un seul checksum est validé lors du téléchargement. La priorité suivante, basée sur l'efficacité de l'algorithme de somme de contrôle, détermine la somme de contrôle validée par le SDK :

1. CRC-32C
2. CRC-32
3. SHA-1
4. SHA-256

Par exemple, si une réponse contient à la fois des sommes de contrôle CRC-32 et SHA-256, seule la somme de contrôle CRC-32 est validée.

Utiliser un client S3 performant : client S3 AWS basé sur CRT

Le client S3 AWS basé sur CRT, basé sur le [AWSCommon Runtime \(CRT\)](#), est un client asynchrone S3 alternatif. [Il transfère des objets vers et depuis Amazon Simple Storage Service \(Amazon S3\) avec des performances et une fiabilité améliorées en utilisant automatiquement l'API de téléchargement en plusieurs parties d'Amazon S3 et les extractions par plage d'octets.](#)

Le client S3 AWS basé sur CRT améliore la fiabilité des transferts en cas de défaillance du réseau. La fiabilité est améliorée en réessayant les différentes parties défectueuses d'un transfert de fichiers sans recommencer le transfert depuis le début.

En outre, le client S3 AWS basé sur CRT offre un regroupement de connexions amélioré et un équilibrage de charge du système de noms de domaine (DNS), ce qui améliore également le débit.

Vous pouvez utiliser le client S3 AWS basé sur CRT à la place du client asynchrone S3 standard du SDK et profiter immédiatement de son débit amélioré.

AWSSComposants du SDK basés sur le CRT

Le client S3 AWS basé sur CRT, décrit dans cette rubrique, et le client HTTP AWS basé sur CRT sont des composants différents du SDK.

Le client S3 AWS basé sur CRT est une implémentation de l'AsyncClientinterface [S3](#) et est utilisé pour travailler avec le service Amazon S3. Il s'agit d'une alternative à l'implémentation Java de l'S3AsyncClientinterface et offre plusieurs avantages.

Le [client HTTP AWS CRT](#) est une implémentation de l'[SdkAsyncHttpClient](#)interface et est utilisé pour les communications HTTP générales. Il s'agit d'une alternative à l'implémentation Netty de l'SdkAsyncHttpClientinterface et offre plusieurs avantages.

Bien que les deux composants utilisent des bibliothèques issues du [AWSCommon Runtime](#), le client S3 AWS basé sur CRT utilise la [bibliothèque aws-c-s 3](#) et prend en charge les fonctionnalités de l'API de [téléchargement partitionné S3](#). Le client HTTP AWS CRT étant destiné à un usage général, il ne prend pas en charge les fonctionnalités de l'API de téléchargement partitionné S3.

Ajoutez des dépendances pour utiliser le client S3 AWS basé sur CRT

Pour utiliser le client S3 AWS basé sur CRT, ajoutez les deux dépendances suivantes à votre fichier de projet Maven. L'exemple montre les versions minimales à utiliser. Recherchez dans le référentiel central Maven les versions les plus récentes des artefacts [s3](#) et [aws-crt](#).

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.68</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.21.16</version>
</dependency>
```

Création d'une instance du client S3 AWS basé sur CRT

Créez une instance du client S3 AWS basé sur CRT avec les paramètres par défaut, comme indiqué dans l'extrait de code suivant.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

Pour configurer le client, utilisez le générateur de clients AWS CRT. Vous pouvez passer du client asynchrone S3 standard au client AWS CRT en modifiant la méthode du générateur.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;

S3AsyncClient s3AsyncClient =
    S3AsyncClient.crtBuilder()
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_WEST_2)
        .targetThroughputInGbps(20.0)
        .minimumPartSizeInBytes(8 * 1025 * 1024L)
        .build();
```

Note

Certains paramètres du générateur standard ne sont peut-être pas actuellement pris en charge dans le générateur client AWS CRT. Obtenez le constructeur standard en appelant `S3AsyncClient#builder()`.

Utiliser le client S3 AWS basé sur CRT

Utilisez le client S3 AWS basé sur CRT pour appeler les opérations d'API Amazon S3. L'exemple suivant illustre les [GetObject](#) opérations [PutObject](#) et disponibles via le AWS SDK for Java.

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

S3AsyncClient s3Client = S3AsyncClient.crtCreate();

// Upload a local file to Amazon S3.
PutObjectResponse putObjectResponse =
    s3Client.putObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
        AsyncRequestBody.fromFile(Paths.get(<FILE_NAME>)))
        .join();

// Download an object from Amazon S3 to a local file.
GetObjectResponse getObjectResponse =
    s3Client.getObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
        AsyncResponseTransformerToFile(Paths.get(<FILE_NAME>)))
        .join();
```

Transférez des fichiers et des répertoires avec Amazon S3 Transfer Manager

Amazon S3 Transfer Manager est un utilitaire de transfert de fichiers open source de haut niveau pour AWS SDK for Java 2.x. Utilisez-le pour transférer des fichiers et des répertoires depuis et vers Amazon Simple Storage Service (Amazon S3).

[Lorsqu'il est construit sur le client S3 AWS basé sur CRT, le S3 Transfer Manager peut tirer parti des améliorations de performances telles que l'API de téléchargement partitionné et les extractions par plage d'octets.](#)

Avec le gestionnaire de transfert S3, vous pouvez également suivre la progression d'un transfert en temps réel et le suspendre pour une exécution ultérieure.

Mise en route

Ajoutez des dépendances à votre fichier de compilation

Pour utiliser le gestionnaire de transfert S3 avec des performances améliorées basées sur le client S3 AWS basé sur CRT, configurez votre fichier de build avec les dépendances suivantes.

- Utilisez la version **2.19.1** ou supérieure du SDK pour Java 2.x.
- Ajoutez l'`s3-transfer-manager`artefact en tant que dépendance.
- Ajoutez l'`aws-crt`artefact en tant que dépendance dans la version **0.20.3 ou supérieure**.

L'exemple de code suivant montre comment configurer les dépendances de votre projet pour Maven.

```
<project>
  <properties>
    <aws.sdk.version>2.19.1</aws.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```



```
</dependency>
</dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3-transfer-manager</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk.crt</groupId>
    <artifactId>aws-crt</artifactId>
    <version>0.20.3</version>
  </dependency>
</dependencies>
</project>
```

[Recherchez dans le référentiel central Maven les versions les plus récentes des artefacts s3-transfer-manager et aws-crt.](#)

Création d'une instance du gestionnaire de transfert S3

L'extrait suivant montre comment créer une `TransferManager` instance [S3](#) avec les paramètres par défaut.

```
S3TransferManager transferManager = S3TransferManager.create();
```

L'exemple suivant montre comment configurer un gestionnaire de transfert S3 avec des paramètres personnalisés. Dans cet exemple, une `AsyncClient` instance [S3 AWS basée sur CRT](#) est utilisée comme client sous-jacent pour le gestionnaire de transfert S3.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .targetThroughputInGbps(20.0)
    .minimumPartSizeInBytes(8 * MB)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

Note

Si la `aws-crt` dépendance n'est pas incluse dans le fichier de compilation, le S3 Transfer Manager est construit sur le client asynchrone S3 standard utilisé dans le SDK pour Java 2.x.

Charger un fichier dans un compartiment S3

L'exemple suivant montre un exemple de téléchargement de fichier ainsi que l'utilisation facultative de [LoggingTransferListener](#), qui enregistre la progression du téléchargement.

Pour charger un fichier sur Amazon S3 à l'aide du gestionnaire de transfert S3, transmettez un [UploadFileRequest](#) objet à `S3TransferManager` la méthode [uploadFile](#).

L'[FileUpload](#) objet renvoyé par la `uploadFile` méthode représente le processus de téléchargement. Une fois la demande terminée, l'[CompletedFileUpload](#) objet contient des informations sur le téléchargement.

```
public String uploadFile(S3TransferManager transferManager, String bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Importations

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
```

```
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

Télécharger un fichier depuis un compartiment S3

L'exemple suivant montre un exemple de téléchargement ainsi que l'utilisation facultative de [LoggingTransferListener](#), qui enregistre la progression du téléchargement.

Pour télécharger un objet depuis un compartiment S3 à l'aide du gestionnaire de transfert S3, créez un [DownloadFileRequest](#) objet et transmettez-le à la méthode [DownloadFile](#).

L'[FileDownload](#) objet renvoyé par la `downloadFile` méthode `S3TransferManager`'s représente le transfert de fichier. Une fois le téléchargement terminé, il [CompletedFileDownload](#) contient l'accès aux informations relatives au téléchargement.

```
public Long downloadFile(S3TransferManager transferManager, String bucketName,
                        String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .destination(Paths.get(downloadedFilePath))
        .build();

    FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
    logger.info("Content length [{}]", downloadResult.response().contentLength());
    return downloadResult.response().contentLength();
}
```

Importations

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
```

```
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

Copier un objet Amazon S3 dans un autre compartiment

L'exemple suivant montre comment copier un objet avec le gestionnaire de transfert S3.

Pour commencer à copier un objet d'un compartiment S3 vers un autre compartiment, créez une [CopyObjectRequest](#) instance de base.

Ensuite, insérez les éléments de base `CopyObjectRequest` dans un [CopyRequest](#) fichier utilisable par le gestionnaire de transfert S3.

L'objet renvoyé par la `copy` méthode `S3TransferManager`'s représente le processus de copie. Une fois le processus de copie terminé, l'[CompletedCopy](#) objet contient les détails de la réponse.

```
public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}
```

Note

Pour effectuer une copie entre régions avec le gestionnaire de transfert S3, activez-le `crossRegionAccessEnabled` sur le générateur de clients S3 AWS basé sur CRT, comme indiqué dans l'extrait suivant.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .crossRegionAccessEnabled(true)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

Importations

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;
```

Charger un répertoire local dans un compartiment S3

L'exemple suivant montre comment télécharger un répertoire local dans S3.

Commencez par appeler la méthode [UploadDirectory](#) de l'`S3TransferManager` instance, en lui transmettant un [UploadDirectoryRequest](#).

L'[DirectoryUpload](#) objet représente le processus de téléchargement, qui génère un [CompletedDirectoryUpload](#) lorsque la demande est terminée. L'`CompletedDirectoryUpload` objet contient des informations sur les résultats du transfert, notamment les fichiers qui n'ont pas pu être transférés.

```
public Integer uploadDirectory(S3TransferManager transferManager,
```

```
URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}
```

Importations

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

Télécharger des objets du compartiment S3 dans un répertoire local

Vous pouvez télécharger les objets d'un compartiment S3 dans un répertoire local, comme illustré dans l'exemple suivant.

Pour télécharger les objets d'un compartiment S3 vers un répertoire local, commencez par appeler la méthode [DownloadDirectory](#) du gestionnaire de transfert, en transmettant un [DownloadDirectoryRequest](#).

L'[DirectoryDownload](#) objet représente le processus de téléchargement, qui génère un [CompletedDirectoryDownload](#) lorsque la demande est terminée.

L'objet `CompletedDirectoryDownload` contient des informations sur les résultats du transfert, notamment les fichiers qui n'ont pas pu être transférés.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

Importations

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;
```

Voir des exemples complets

[GitHub contient le code complet](#) pour tous les exemples de cette page.

Travailler avec Amazon Simple Notification Service

Avec Amazon Simple Notification Service, vous pouvez facilement transmettre des messages de notification en temps réel à partir de vos applications vers les abonnés via plusieurs canaux de communication. Cette rubrique décrit comment exécuter certaines des fonctions de base de Amazon SNS.

Créer une rubrique

Une rubrique est un regroupement logique de canaux de communication qui définit les systèmes auxquels envoyer un message, par exemple, en diffusant un message vers AWS Lambda et un webhook HTTP. Vous envoyez à Amazon SNS des messages qui seront distribués via les canaux définis dans la rubrique. Les messages sont alors disponibles pour les abonnés.

Pour créer un sujet, créez d'abord un [CreateTopicRequest](#) objet, avec le nom du sujet défini à l'aide de la `name()` méthode du générateur. Ensuite, envoyez l'objet de la demande à Amazon SNS en utilisant la méthode `createTopic()` du [SnsClient](#). Vous pouvez capturer le résultat de cette demande sous forme d'[CreateTopicResponse](#) objet, comme illustré dans l'extrait de code suivant.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {

    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();
```



```
        result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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

Listez vos Amazon SNS sujets

Pour récupérer la liste de vos Amazon SNS sujets existants, créez un [ListTopicsRequest](#) objet. Ensuite, envoyez l'objet de la demande à Amazon SNS en utilisant la méthode `listTopics()` du `SnsClient`. Vous pouvez capturer le résultat de cette demande sous forme d'[ListTopicsResponse](#) objet.

L'extrait de code suivant imprime le code d'état HTTP de la demande et une liste d'ARN (Amazon Resource Name) pour vos rubriques Amazon SNS.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
public static void listSNSTopics(SnsClient snsClient) {

    try {
        ListTopicsRequest request = ListTopicsRequest.builder()
            .build();

        ListTopicsResponse result = snsClient.listTopics(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode() +
            "\n\nTopics\n\n" + result.topics());
    }
}
```

```
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

Abonner un point de terminaison à une rubrique

Après avoir créé une rubrique, vous pouvez configurer les canaux de communication qui deviendront des points de terminaison pour cette rubrique. Les messages sont distribués à ces points de terminaison après leur réception par Amazon SNS.

Pour configurer un canal de communication en tant que point de terminaison pour une rubrique, abonnez ce point de terminaison à la rubrique. Pour commencer, créez un [SubscribeRequest](#) objet. Spécifiez le canal de communication (par exemple, `lambda` ou `email`) en tant que `protocol()`. Définissez l'emplacement `endpoint()` de sortie approprié (par exemple, l'ARN d'une Lambda fonction ou une adresse e-mail), puis définissez l'ARN du sujet auquel vous souhaitez vous abonner en tant que `topicArn()`. Envoyez l'objet de demande à Amazon SNS en utilisant la `subscribe()` méthode du `SnsClient`. Vous pouvez capturer le résultat de cette demande sous forme d'[SubscribeResponse](#) objet.

L'extrait de code suivant montre comment abonner une adresse e-mail à une rubrique.

Importations

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.SnsException;  
import software.amazon.awssdk.services.sns.model.SubscribeRequest;  
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

Code

```
public static void subEmail(SnsClient snsClient, String topicArn, String email) {  
  
    try {  
        SubscribeRequest request = SubscribeRequest.builder()  
            .protocol("email")  
            .endpoint(email)
```

```
        .returnSubscriptionArn(true)
        .topicArn(topicArn)
        .build();

    SubscribeResponse result = snsClient.subscribe(request);
    System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
        + "Status is " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Publier un message dans une rubrique

Après avoir configuré une rubrique et un ou plusieurs points de terminaison, vous pouvez y publier un message. Pour commencer, créez un [PublishRequest](#) objet. Spécifiez le message () à envoyer et l'ARN de la rubrique (topicArn ()) à laquelle l'envoyer. Ensuite, envoyez l'objet de la demande à Amazon SNS en utilisant la méthode publish () du SnsClient. Vous pouvez capturer le résultat de cette demande sous forme d'[PublishResponse](#) objet.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();
```

```
        PublishResponse result = snsClient.publish(request);
        System.out.println(result.messageId() + " Message sent. Status is " +
            result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Désabonner un point de terminaison à une rubrique

Vous pouvez supprimer les canaux de communication configurés en tant que points de terminaison pour une rubrique. Après cela, la rubrique elle-même continue d'exister et de diffuser des messages à tous les autres points de terminaison configurés pour cette rubrique.

Pour supprimer un canal de communication en tant que point de terminaison pour une rubrique, annulez l'abonnement du point de terminaison à la rubrique. Pour commencer, créez un [UnsubscribeRequest](#) objet et définissez l'ARN du sujet dont vous souhaitez vous désabonner comme étant les `subscriptionArn()`. Ensuite, envoyez l'objet de demande à SNS en utilisant la méthode `unsubscribe()` du `SnsClient`. Vous pouvez capturer le résultat de cette demande sous forme d'[UnsubscribeResponse](#) objet.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

Code

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {

    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();
```

```
UnsubscribeResponse result = snsClient.unsubscribe(request);

System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
    + "\n\nSubscription was removed for " + request.subscriptionArn());

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

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

Supprimer une rubrique

Pour supprimer un Amazon SNS sujet, créez d'abord un [DeleteTopicRequest](#) objet avec l'ARN du sujet défini comme `topicArn()` méthode dans le générateur. Ensuite, envoyez l'objet de la demande à Amazon SNS en utilisant la méthode `deleteTopic()` du `SnsClient`. Vous pouvez capturer le résultat de cette demande sous forme d'[DeleteTopicResponse](#) objet, comme illustré dans l'extrait de code suivant.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {

    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());
    }
}
```

```
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

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

Pour plus d'informations, consultez le [Guide du développeur Amazon Simple Notification Service](#).

Travaillez avec Amazon Simple Queue Service

Cette section fournit des exemples de programmation à l'[Amazon Simple Queue Service](#) aide de la version AWS SDK for Java 2.x.

Les exemples suivants 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

- [Travailler avec les files d'attente de Amazon Simple Queue Service messages](#)
- [Envoyer, recevoir et supprimer Amazon Simple Queue Service des messages](#)

Travailler avec les files d'attente de Amazon Simple Queue Service messages

Une file d'attente de messages est le conteneur logique utilisé pour envoyer en toute fiabilité des messages dans Amazon Simple Queue Service. 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 [guide du Amazon Simple Queue Service développeur](#).

Cette rubrique explique comment créer, répertorier, supprimer une file d'attente Amazon Simple Queue Service et obtenir son URL à l'aide du kit AWS SDK for Java.

La `sqsClient` variable utilisée dans les exemples suivants peut être créée à partir de l'extrait de code suivant.

```
SqsClient sqsClient = SqsClient.create();
```

Lorsque vous créez un `SqsClient` utilisant la `create()` méthode statique, le SDK configure la région en utilisant la chaîne de fournisseurs de régions par [défaut et les informations d'identification en utilisant la chaîne de fournisseurs](#) d'informations [d'identification par défaut](#).

Créer une file d'attente

Utilisez la `SqsClient`'s `createQueue` méthode et fournissez un [CreateQueueRequest](#) objet qui décrit les paramètres de la file d'attente, comme indiqué dans l'extrait de code suivant.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
    .queueName(queueName)
    .build();

sqsClient.createQueue(createQueueRequest);
```

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

Répertorier les files d'attente

Pour répertorier les Amazon Simple Queue Service files d'attente pour votre compte, appelez la `SqsClient`'s `listQueues` méthode avec un [ListQueuesRequest](#) objet.

Lorsque vous utilisez la forme de la [listQueues](#) méthode qui ne prend aucun paramètre, le service renvoie toutes les files d'attente, jusqu'à 1 000 files d'attente.

Vous pouvez fournir un préfixe de nom de file d'attente à l'[ListQueuesRequest](#) objet afin de limiter les résultats aux files d'attente correspondant à ce préfixe, comme indiqué dans le code suivant.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);

    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

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

Obtenir l'URL d'une file d'attente

Le code suivant montre comment obtenir l'URL d'une file d'attente en appelant la `SqsClient`'s `getQueueUrl` méthode avec un [GetQueueUrlRequest](#) objet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
GetQueueUrlResponse getQueueUrlResponse =
```



```
sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();
return queueUrl;
```

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

Suppression d'une file d'attente

Fournissez l'[URL](#) de la file d'attente vers l'[DeleteQueueRequest](#) objet. Appelez ensuite la `SqsClient`'s `deleteQueue` méthode pour supprimer une file d'attente comme indiqué dans le code suivant.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

    try {

        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();

        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

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

En savoir plus

- [CreateQueue](#) dans la référence de Amazon Simple Queue Service l'API
- [GetQueueUrl](#) dans la référence de Amazon Simple Queue Service l'API
- [ListQueues](#) dans la référence de Amazon Simple Queue Service l'API
- [DeleteQueue](#) dans la référence de Amazon Simple Queue Service l'API

Envoyer, recevoir et supprimer Amazon Simple Queue Service des messages

Un message est une portion de données qui peut être envoyée et reçue par des composants distribués. Les messages sont toujours livrés à l'aide d'une [file d'attente SQS](#).

La `sqsClient` variable utilisée dans les exemples suivants peut être créée à partir de l'extrait de code suivant.

```
SqsClient sqsClient = SqsClient.create();
```

Lorsque vous créez un `SqsClient` utilisant la `create()` méthode statique, le SDK configure la région en utilisant la chaîne de fournisseurs de régions par [défaut et les informations d'identification en utilisant la chaîne de fournisseurs](#) d'informations [d'identification par défaut](#).

Envoyer un message

Ajoutez un seul message à une Amazon Simple Queue Service file d'attente en appelant la `sendMessage` méthode `SqsClient` client. Fournissez un [SendMessageRequest](#) objet contenant l'[URL](#) de la file d'attente, le corps du message et la valeur de délai facultative (en secondes).

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
sqscClient.sendMessage(SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody("Hello world!")
    .delaySeconds(10)
    .build());

sqscClient.sendMessage(sendMsgRequest);
```

Envoyer plusieurs messages dans une demande

Envoyez plusieurs messages dans une seule demande à l'aide de la méthode `sendMessageBatch` d' `SqsClient` . Cette méthode utilise un [SendMessageBatchRequest](#) qui contient l'URL de la file d'attente et une liste de messages à envoyer. (Chaque message est un [SendMessageBatchRequestEntry](#).) Vous pouvez aussi retarder l'envoi d'un message spécifique en définissant une valeur de retard sur le message.

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
    .build();
sqscClient.sendMessageBatch(sendMessageBatchRequest);
```

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

Extraction des messages

Récupérez tous les messages qui sont actuellement dans la file d'attente en appelant la méthode `receiveMessage` d' `SqsClient` . Cette méthode utilise un [ReceiveMessageRequest](#) qui contient l'URL de la file d'attente. Vous pouvez également spécifier le nombre maximal de messages à renvoyer. Les messages sont renvoyés sous la forme d'une liste d'objets [Message](#).

Importations

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .numberOfMessages(5)
        .build();
    List<Message> messages =
sqsClient.receiveMessage(receiveMessageRequest).messages();
    return messages;
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

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

Supprimer un message après réception

Après avoir reçu un message et traité son contenu, supprimez-le de la file d'attente en envoyant le descripteur de réception du message et l'URL de la file d'attente à la `SqsClient` 's [deleteMessage](#) méthode.

Importations

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
}
```

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

Plus d'informations

- [Comment fonctionnent les Amazon Simple Queue Service files d'attente](#) dans le guide du Amazon Simple Queue Service développeur
- [SendMessage](#) dans la référence de Amazon Simple Queue Service l'API
- [SendMessageBatch](#) dans la référence de Amazon Simple Queue Service l'API
- [ReceiveMessage](#) dans la référence de Amazon Simple Queue Service l'API
- [DeleteMessage](#) dans la référence de Amazon Simple Queue Service l'API

Travailler avec Amazon Transcribe

L'exemple suivant montre comment le streaming bidirectionnel fonctionne avec Amazon Transcribe. Le streaming bidirectionnel implique qu'un flux de données soit à la fois envoyé au service et reçu en temps réel. L'exemple utilise la transcription de flux en streaming Amazon Transcribe pour envoyer un flux audio et recevoir un flux de texte transcrit en temps réel.

Consultez la section [Transcription en streaming](#) dans le guide du Amazon Transcribe développeur pour en savoir plus sur cette fonctionnalité.

Consultez [Getting Started](#) dans le guide du Amazon Transcribe développeur pour commencer à utiliser Amazon Transcribe.

Configurez le microphone

Ce code utilise le package `javax.sound.sampled` pour créer un flux en streaming audio à partir d'un périphérique d'entrée.

Code

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;

public class Microphone {

    public static TargetDataLine get() throws Exception {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
        DataLine.Info datalineInfo = new DataLine.Info(TargetDataLine.class, format);

        TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(datalineInfo);
        dataLine.open(format);

        return dataLine;
    }
}
```

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

Création d'un éditeur

Ce code met en œuvre un éditeur qui publie des données audio à partir du flux audio Amazon Transcribe.

Code

```
package com.amazonaws.transcribe;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
```

```
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.transcribestreaming.model.AudioEvent;
import software.amazon.awssdk.services.transcribestreaming.model.AudioStream;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException;

public class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;

    public AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
        s.onSubscribe(new SubscriptionImpl(s, inputStream));
    }

    private class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;

        private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);
        }
    }
}
```

```
    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (TranscribeStreamingException e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {

}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}
```



```
private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

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

Créez le client et lancez le stream

Dans la méthode principale, créez une demande d'objet, démarrez le flux d'entrée audio et instanciez l'éditeur avec l'entrée audio.

Vous devez également créer un [StartStreamTranscriptionResponseHandler](#) pour spécifier comment gérer la réponse de Amazon Transcribe.

Ensuite, utilisez la `startStreamTranscription` méthode `TranscribeStreamingAsyncClient`'s pour démarrer le streaming bidirectionnel.

Importations

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioInputStream;
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException ;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionRequest;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponseHandler;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
```

Code

```
public static void convertAudio(TranscribeStreamingAsyncClient client) throws
Exception {

    try {

        StartStreamTranscriptionRequest request =
StartStreamTranscriptionRequest.builder()
            .mediaEncoding(MediaEncoding.PCM)
            .languageCode(LanguageCode.EN_US)
            .mediaSampleRateHertz(16_000).build();

        TargetDataLine mic = Microphone.get();
        mic.start();

        AudioStreamPublisher publisher = new AudioStreamPublisher(new
AudioInputStream(mic));

        StartStreamTranscriptionResponseHandler response =
            StartStreamTranscriptionResponseHandler.builder().subscriber(e -> {
                TranscriptEvent event = (TranscriptEvent) e;
                event.transcript().results().forEach(r ->
r.alternatives().forEach(a -> System.out.println(a.transcript())));
            }).build();

        // Keeps Streaming until you end the Java program
        client.startStreamTranscription(request, publisher, response);

    } catch (TranscribeStreamingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

En savoir plus

- [Comment cela fonctionne](#) dans le guide du Amazon Transcribe développeur.
- [Pour démarrer avec le streaming audio](#), consultez le guide du Amazon Transcribe développeur.

Exemples de code du SDK pour Java 2.x

Les exemples de code présentés dans cette rubrique vous montrent comment utiliser le AWS SDK for Java 2.x with AWS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Les Exemples de services croisés sont des exemples d'applications fonctionnant sur plusieurs Services AWS.

Exemples

- [Actions et scénarios utilisant le SDK for Java 2.x](#)
- [Exemples multiservices utilisant le SDK pour Java 2.x](#)

Actions et scénarios utilisant le SDK for Java 2.x

Les exemples de code suivants montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with Services AWS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Services

- [Exemples d'API Gateway utilisant le SDK pour Java 2.x](#)
- [Exemples d'Application Auto Scaling à l'aide du SDK pour Java 2.x](#)
- [Exemples d'applications Recovery Controller utilisant le SDK for Java 2.x](#)

- [Exemples d'Aurora utilisant le SDK pour Java 2.x](#)
- [Exemples d'Auto Scaling utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon Bedrock utilisant le SDK pour Java 2.x](#)
- [Exemples d'exécution Amazon Bedrock utilisant le SDK pour Java 2.x](#)
- [CloudFront exemples d'utilisation du SDK pour Java 2.x](#)
- [CloudWatch exemples d'utilisation du SDK pour Java 2.x](#)
- [CloudWatch Exemples d'événements utilisant le SDK pour Java 2.x](#)
- [CloudWatch Exemples de journaux utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon Cognito Identity utilisant le SDK pour Java 2.x](#)
- [Exemples de fournisseurs d'identité Amazon Cognito utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon Comprehend utilisant le SDK pour Java 2.x](#)
- [Exemples DynamoDB utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon EC2 utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon ECS utilisant le SDK pour Java 2.x](#)
- [Elastic Load Balancing - Exemples de version 2 utilisant le SDK pour Java 2.x](#)
- [MediaStore exemples d'utilisation du SDK pour Java 2.x](#)
- [OpenSearch Exemples de services utilisant le SDK pour Java 2.x](#)
- [EventBridge exemples d'utilisation du SDK pour Java 2.x](#)
- [Exemples de prévisions utilisant le SDK pour Java 2.x](#)
- [AWS Glue exemples d'utilisation du SDK pour Java 2.x](#)
- [HealthImaging exemples d'utilisation du SDK pour Java 2.x](#)
- [Exemples d'IAM utilisant le SDK pour Java 2.x](#)
- [AWS IoT exemples d'utilisation du SDK pour Java 2.x](#)
- [AWS IoT data exemples d'utilisation du SDK pour Java 2.x](#)
- [Exemples d'Amazon Keyspaces utilisant le SDK pour Java 2.x](#)
- [Exemples Kinesis utilisant le SDK pour Java 2.x](#)
- [AWS KMS exemples d'utilisation du SDK pour Java 2.x](#)
- [Exemples Lambda utilisant le SDK pour Java 2.x](#)
- [MediaConvert exemples d'utilisation du SDK pour Java 2.x](#)
- [Exemples de Migration Hub utilisant le SDK pour Java 2.x](#)

- [Exemples Amazon Personalize à l'aide du SDK pour Java 2.x](#)
- [Exemples d'événements Amazon Personalize à l'aide du SDK pour Java 2.x](#)
- [Exemples d'Amazon Personalize Runtime à l'aide du SDK pour Java 2.x](#)
- [Exemples d'Amazon Pinpoint utilisant le SDK pour Java 2.x](#)
- [Exemples d'API SMS et vocales Amazon Pinpoint à l'aide du SDK pour Java 2.x](#)
- [Exemples d'Amazon Polly utilisant le SDK pour Java 2.x](#)
- [Exemples Amazon RDS utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon Redshift utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon Rekognition utilisant le SDK pour Java 2.x](#)
- [Exemples d'enregistrement de domaine Route 53 à l'aide du SDK for Java 2.x](#)
- [Exemples d'Amazon S3 utilisant le SDK pour Java 2.x](#)
- [Exemples de S3 Glacier utilisant le SDK pour Java 2.x](#)
- [SageMaker exemples d'utilisation du SDK pour Java 2.x](#)
- [Exemples de Secrets Manager utilisant le SDK pour Java 2.x](#)
- [Exemples Amazon SES utilisant le SDK pour Java 2.x](#)
- [Exemples d'API Amazon SES v2 utilisant le SDK pour Java 2.x](#)
- [Exemples Amazon SNS utilisant le SDK pour Java 2.x](#)
- [Exemples Amazon SQS utilisant le SDK pour Java 2.x](#)
- [Exemples de Step Functions utilisant le SDK pour Java 2.x](#)
- [AWS STS exemples d'utilisation du SDK pour Java 2.x](#)
- [AWS Support exemples d'utilisation du SDK pour Java 2.x](#)
- [Exemples de Systems Manager utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon Textract utilisant le SDK pour Java 2.x](#)
- [Exemples d'Amazon Transcribe utilisant le SDK pour Java 2.x](#)

Exemples d'API Gateway utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK for Java 2.x with API Gateway.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateDeployment

L'exemple de code suivant montre comment utiliser `CreateDeployment`.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

    try {
        CreateDeploymentRequest request = CreateDeploymentRequest.builder()
            .restApiId(restApiId)
            .description("Created using the AWS API Gateway Java API")
            .stageName(stageName)
            .build();

        CreateDeploymentResponse response =
apiGateway.createDeployment(request);
        System.out.println("The id of the deployment is " + response.id());
        return response.id();
    }
}
```

```
    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDeployment](#) à la section Référence des AWS SDK for Java 2.x API.

CreateRestApi

L'exemple de code suivant montre comment utiliser `CreateRestApi`.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createAPI(ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

    try {
        CreateRestApiRequest request = CreateRestApiRequest.builder()
            .cloneFrom(restApiId)
            .description("Created using the Gateway Java API")
            .name(restApiName)
            .build();

        CreateRestApiResponse response = apiGateway.createRestApi(request);
        System.out.println("The id of the new api is " + response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }
```

- Pour plus de détails sur l'API, reportez-vous [CreateRestApi](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteDeployment

L'exemple de code suivant montre comment utiliser `DeleteDeployment`.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String  
restApiId, String deploymentId) {  
  
    try {  
        DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()  
            .restApiId(restApiId)  
            .deploymentId(deploymentId)  
            .build();  
  
        apiGateway.deleteDeployment(request);  
        System.out.println("Deployment was deleted");  
  
    } catch (ApiGatewayException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDeployment](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteRestApi

L'exemple de code suivant montre comment utiliser `DeleteRestApi`.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteAPI(ApiGatewayClient apiGateway, String restApiId) {  
  
    try {  
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()  
            .restApiId(restApiId)  
            .build();  
  
        apiGateway.deleteRestApi(request);  
        System.out.println("The API was successfully deleted");  
  
    } catch (ApiGatewayException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteRestApi](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'Application Auto Scaling à l'aide du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de l' AWS SDK for Java 2.x application Application Auto Scaling.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

DeleteScalingPolicy

L'exemple de code suivant montre comment utiliser DeleteScalingPolicy.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
```

```
import
software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import
software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DisableDynamoDBAutoScaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).\s
                policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];
```

```
deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
}

public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
    try {
        DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
            .policyName(policyName)
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        appAutoScalingClient.deleteScalingPolicy(delSPRequest);
        System.out.println(policyName + " was deleted successfully.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the scaling policy was deleted
public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceId(tableId)
        .build();

    DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}
```

```
public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    try {
        DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        appAutoScalingClient.deregisterScalableTarget(targetRequest);
        System.out.println("The scalable target was deregistered.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteScalingPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

RegisterScalableTarget

L'exemple de code suivant montre comment utiliser `RegisterScalableTarget`.

SDK pour Java 2.x

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <roleARN> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).
                roleARN - The ARN of the role that has ApplicationAutoScaling
permissions.
                policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table, which
is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }
}
```

```
public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
    try {
        RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
            .serviceNamespace(ns)
            .scalableDimension(tableWCUs)
            .resourceId(tableId)
            .roleARN(roleARN)
            .minCapacity(5)
            .maxCapacity(10)
            .build();

        appAutoScalingClient.registerScalableTarget(targetRequest);
        System.out.println("You have registered " + tableId);

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the target was created.
public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}

// Configure a scaling policy.
public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
    // Check if the policy exists before creating a new one.
```



```
DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
    .serviceNamespace(ns)
    .resourceId(tableId)
    .scalableDimension(tableWCUs)
    .build());

if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
    // If policies exist, consider updating an existing policy instead of
creating a new one.
    System.out.println("Policy already exists. Consider updating it
instead.");
    List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
    for (ScalingPolicy pol : polList) {
        System.out.println("Policy name:" +pol.policyName());
    }
} else {
    // If no policies exist, proceed with creating a new policy.
    PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
    .build();

    TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
    .predefinedMetricSpecification(specification)
    .targetValue(50.0)
    .scaleInCooldown(60)
    .scaleOutCooldown(60)
    .build();

    PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
    .targetTrackingScalingPolicyConfiguration(policyConfiguration)
    .serviceNamespace(ns)
    .scalableDimension(tableWCUs)
    .resourceId(tableId)
    .policyName(policyName)
    .policyType(PolicyType.TARGET_TRACKING_SCALING)
    .build();

    try {
```

```
        appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
        System.out.println("You have successfully created a scaling policy
for an Application Auto Scaling scalable target");
    } catch (ApplicationAutoScalingException e) {
        System.err.println("Error: " + e.awsErrorDetails().errorMessage());
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [RegisterScalableTarget](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'applications Recovery Controller utilisant le SDK for Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with Application Recovery Controller.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques


- [Actions](#)

Actions

GetRoutingControlState

L'exemple de code suivant montre comment utiliser `GetRoutingControlState`.

SDK pour Java 2.x

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetRoutingControlState](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateRoutingControlState

L'exemple de code suivant montre comment utiliser `UpdateRoutingControlState`.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn,
    String routingControlState) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region()))
                .build();
            return client.updateRoutingControlState(
                UpdateRoutingControlStateRequest.builder()

.routingControlArn(routingControlArn).routingControlState(routingControlState).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateRoutingControlState](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'Aurora utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Aurora.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Aurora

Les exemples de code suivants montrent comment bien démarrer avec Aurora.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
```

```
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

describeClusters(rdsClient);
rdsClient.close();
}

public static void describeClusters(RdsClient rdsClient) {
    DescribeDBClustersIterable clustersIterable =
rdsClient.describeDBClustersPaginator();
    clustersIterable.stream()
        .flatMap(r -> r.dbClusters().stream())
        .forEach(cluster -> System.out
            .println("Database name: " + cluster.databaseName() + " Arn
= " + cluster.dbClusterArn()));
}
}
```

- Pour plus d'informations sur l'API, consultez [DescribeDBClusters](#) dans la Référence d'API AWS SDK for Java 2.x .

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateDBCluster

L'exemple de code suivant montre comment utiliser CreateDBCluster.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
    String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus d'informations sur l'API, consultez [CreateDBCluster](#) dans AWS SDK for Java 2.x API Reference.

CreateDBClusterParameterGroup

L'exemple de code suivant montre comment utiliser `CreateDBClusterParameterGroup`.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
    String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, voir [CreateDB ClusterParameterGroup](#) dans la référence des AWS SDK for Java 2.x API.

CreateDBClusterSnapshot

L'exemple de code suivant montre comment utiliser `CreateDBClusterSnapshot`.

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
```



```
try {
    CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

    CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
    System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, voir [CreateDB ClusterSnapshot](#) dans la référence des AWS SDK for Java 2.x API.

CreateDBInstance

L'exemple de code suivant montre comment utiliser `CreateDBInstance`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```

```
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus d'informations sur l'API, consultez [CreateDBInstance](#) dans AWS SDK for Java 2.x API Reference.

DeleteDBCluster

L'exemple de code suivant montre comment utiliser `DeleteDBCluster`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
```

```

        .skipFinalSnapshot(true)
        .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Pour plus d'informations sur l'API, consultez [DeleteDBCluster](#) dans AWS SDK for Java 2.x API Reference.

DeleteDBClusterParameterGroup

L'exemple de code suivant montre comment utiliser `DeleteDBClusterParameterGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();

```

```

        int listSize = instanceList.size();
        didFind = false;
        int index = 1;
        for (DBInstance instance : instanceList) {
            instanceARN = instance.dbInstanceArn();
            if (instanceARN.compareTo(clusterDBARN) == 0) {
                System.out.println(clusterDBARN + " still exists");
                didFind = true;
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.
                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

    DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
        .builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .build();

    rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
    System.out.println(dbClusterGroupName + " was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

```

- Pour plus de détails sur l'API, voir [DeleteDB ClusterParameterGroup dans la référence](#) des AWS SDK for Java 2.x API.

DeleteDBInstance

L'exemple de code suivant montre comment utiliser `DeleteDBInstance`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus d'informations sur l'API, consultez [DeleteDBInstance](#) dans la Référence d'API AWS SDK for Java 2.x .

DescribeDBClusterParameterGroups

L'exemple de code suivant montre comment utiliser `DescribeDBClusterParameterGroups`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, voir [DescribeDB ClusterParameterGroups dans le Guide](#) de référence des AWS SDK for Java 2.x API.

DescribeDBClusterParameters

L'exemple de code suivant montre comment utiliser `DescribeDBClusterParameters`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
            }
        }
    }
}
```

```
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, voir [DescribeDB ClusterParameters dans le Guide](#) de référence des AWS SDK for Java 2.x API.

DescribeDBClusterSnapshots

L'exemple de code suivant montre comment utiliser `DescribeDBClusterSnapshots`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
    String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();
```



```
        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
            List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }

        System.out.println("The Snapshot is available!");

    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, voir [DescribeDB ClusterSnapshots dans le Guide](#) de référence des AWS SDK for Java 2.x API.

DescribeDBClusters

L'exemple de code suivant montre comment utiliser `DescribeDBClusters`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
                System.out.println("*** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
}
```

- Pour plus d'informations sur l'API, consultez [DescribeDBClusters](#) dans la Référence d'API AWS SDK for Java 2.x .

DescribeDBEngineVersions

L'exemple de code suivant montre comment utiliser `DescribeDBEngineVersions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }
    }
}
```

```
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, voir [DescribeDB EngineVersions dans le Guide](#) de référence des AWS SDK for Java 2.x API.

DescribeDBInstances

L'exemple de code suivant montre comment utiliser `DescribeDBInstances`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Waits until the database instance is available.  
public static void waitForInstanceReady(RdsClient rdsClient, String  
dbClusterIdentifier) {  
    boolean instanceReady = false;  
    String instanceReadyStr;  
    System.out.println("Waiting for instance to become available.");  
    try {  
        DescribeDbClustersRequest instanceRequest =  
DescribeDbClustersRequest.builder()  
            .dbClusterIdentifier(dbClusterIdentifier)  
            .build();  
  
        while (!instanceReady) {  
            DescribeDbClustersResponse response =  
rdsClient.describeDBClusters(instanceRequest);  
            List<DBCluster> clusterList = response.dbClusters();  
            for (DBCluster cluster : clusterList) {  
                instanceReadyStr = cluster.status();  
            }  
        }  
    }  
}
```

```
        if (instanceReadyStr.contains("available")) {
            instanceReady = true;
        } else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}
System.out.println("Database cluster is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus d'informations sur l'API, consultez [DescribeDBInstances](#) dans la Référence d'API AWS SDK for Java 2.x .

DescribeOrderableDBInstanceOptions

L'exemple de code suivant montre comment utiliser `DescribeOrderableDBInstanceOptions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();
```

```
DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
List<DBEngineVersion> engines = response.dbEngineVersions();

// Get all DBEngineVersion objects.
for (DBEngineVersion engineOb : engines) {
    System.out.println("The name of the DB parameter group family for
the database engine is "
        + engineOb.dbParameterGroupFamily());
    System.out.println("The name of the database engine " +
engineOb.engine());
    System.out.println("The version number of the database engine " +
engineOb.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, voir [DescribeOrderableDB InstanceOptions](#) dans le Guide de référence des AWS SDK for Java 2.x API.

ModifyDBClusterParameterGroup

L'exemple de code suivant montre comment utiliser `ModifyDBClusterParameterGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
```

```
DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .maxRecords(20)
    .build();

List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
    .dbClusterParameterGroups();
for (DBClusterParameterGroup group : groups) {
    System.out.println("The group name is " +
group.dbClusterParameterGroupName());
    System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, voir [ModifyDB ClusterParameterGroup dans la référence](#) des AWS SDK for Java 2.x API.

Scénarios

Démarrage avec les clusters de base de données

L'exemple de code suivant illustre comment :

- Créez un groupe de paramètres pour le cluster de base de données Aurora personnalisé et définissez des valeurs pour les paramètres.
- Créez un cluster de base de données qui utilise le groupe de paramètres.
- Créez une instance de base de données qui contient une base de données.
- Prenez un instantané du cluster de base de données, puis nettoyez les ressources.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
 * services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
```



```

* 11. Creates a database instance in the cluster.
* 12. Waits for DB instance to be ready.
* 13. Creates a snapshot.
* 14. Waits for DB snapshot to be ready.
* 15. Deletes the DB cluster.
* 16. Deletes the DB cluster group.
*/
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <dbClusterGroupName> <dbParameterGroupFamily>
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
<dbSnapshotIdentifier><secretName>"
            +
            "Where:\n" +
            "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
            "    dbParameterGroupFamily - The DB cluster parameter group family
name (for example, aurora-mysql5.7). \n"
            +
            "    dbInstanceClusterIdentifier - The instance cluster identifier
value.\n" +
            "    dbInstanceIdentifier - The database instance identifier.\n" +
            "    dbName - The database name.\n" +
            "    dbSnapshotIdentifier - The snapshot identifier.\n" +
            "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\`\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];

```

```
String secretName = args[6];

// Retrieve the database credentials using AWS Secrets Manager.
Gson gson = new Gson();
User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
String username = user.getUsername();
String userPassword = user.getPassword();

Region region = Region.US_WEST_2;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Aurora example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Return a list of the available DB engines");
describeDBEngines(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a custom parameter group");
createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBClusterParas(rdsClient, dbClusterGroupName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("6. Display the updated parameter value");
describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an Aurora DB cluster database");
String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
    username, userPassword);
System.out.println("The ARN of the cluster is " + arnClusterVal);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Wait for DB instance to be ready");
waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of instance classes available for the
selected engine");
String instanceClass = getListInstanceClasses(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a database instance in the cluster.");
String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
    instanceClass);
System.out.println("The ARN of the database is " + clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB instance to be ready");
waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Create a snapshot");
```

```
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
```

```

        .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
        throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

        DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
            .builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .build();

```

```
        rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
        System.out.println(dbClusterGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .skipFinalSnapshot(true)
            .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
    String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
            List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }

        System.out.println("The Snapshot is available!");

    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                }
            }
        }
    }
}
```



```

        Thread.sleep(sleepTime * 1000);
    }
}
}
System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String getListInstanceClasses(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
        .builder()

```

```

        .engine("aurora-mysql")
        .maxRecords(20)
        .build();

DescribeOrderableDbInstanceOptionsResponse response = rdsClient
    .describeOrderableDBInstanceOptions(optionsRequest);
List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
String instanceClass = "";
for (OrderableDBInstanceOption instanceOption : instanceOptions) {
    instanceClass = instanceOption.dbInstanceClass();
    System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
    System.out.println("The engine version is " +
instanceOption.engineVersion());
}
return instanceClass;

} catch (RdsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {

```

```

        instanceReady = true;
    } else {
        System.out.print(".");
        Thread.sleep(sleepTime * 1000);
    }
}
}
System.out.println("Database cluster is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {

```

```
DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("aurora-mysql")
    .build();

DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
List<DBEngineVersion> dbEngines = response.dbEngineVersions();
for (DBEngineVersion dbEngine : dbEngines) {
    System.out.println("The engine version is " +
dbEngine.engineVersion());
    System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Modify the auto_increment_offset parameter.
public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dClusterGroupName)
            .parameters(paraList)
            .build();

        ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
        System.out.println(
```

```
        "The parameter group " + response.dbClusterParameterGroupName()
+ " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("**** The parameter name is " + paraName);
                System.out.println("**** The parameter value is " +
para.parameterValue());
                System.out.println("**** The parameter data type is " +
para.dataType());
                System.out.println("**** The parameter description is " +
para.description());
            }
        }
    }
}
```

```
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
```

```
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateDBCluster](#)
 - [Créer une base de données ClusterParameterGroup](#)
 - [Créer une base de données ClusterSnapshot](#)
 - [CreateDBInstance](#)
 - [DeleteDBCluster](#)
 - [Supprimer B ClusterParameterGroup](#)
 - [DeleteDBInstance](#)
 - [Décrit B ClusterParameterGroups](#)
 - [Décrit B ClusterParameters](#)
 - [Décrit B ClusterSnapshots](#)
 - [DescribeDBClusters](#)
 - [Décrit B EngineVersions](#)
 - [DescribeDBInstances](#)
 - [DescribeOrderableDB InstanceOptions](#)
 - [Modifier la base de données ClusterParameterGroup](#)

Exemples d'Auto Scaling utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK for Java 2.x with Auto Scaling.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Auto Scaling

Les exemples de code suivants montrent comment commencer à utiliser Auto Scaling.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
```

```
        System.out.println("Group Name: " + group.autoScalingGroupName());
        System.out.println("Group ARN: " + group.autoScalingGroupARN());
    });
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAutoScalingGroups](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateAutoScalingGroup

L'exemple de code suivant montre comment utiliser `CreateAutoScalingGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
```

```
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
<vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
        String groupName,
```

```
        String launchTemplateName,
        String vpcZoneId) {

    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter

            .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateAutoScalingGroup](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteAutoScalingGroup

L'exemple de code suivant montre comment utiliser `DeleteAutoScalingGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName>

            Where:
                groupName - The name of the Auto Scaling group.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
```

```
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAutoScalingGroup](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeAutoScalingGroups

L'exemple de code suivant montre comment utiliser `DescribeAutoScalingGroups`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <groupName>

                Where:
                groupName - The name of the Auto Scaling group.
        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }
}
```

```
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient, String
groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());


                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAutoScalingGroups](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeAutoScalingInstances

L'exemple de code suivant montre comment utiliser `DescribeAutoScalingInstances`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAutoScalingInstances](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeScalingActivities

L'exemple de code suivant montre comment utiliser `DescribeScalingActivities`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response = autoScalingClient
            .describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity : activities) {
            System.out.println("The activity Id is " + activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeScalingActivities](#) à la section Référence des AWS SDK for Java 2.x API.

DisableMetricsCollection

L'exemple de code suivant montre comment utiliser `DisableMetricsCollection`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DisableMetricsCollection](#) à la section Référence des AWS SDK for Java 2.x API.

EnableMetricsCollection

L'exemple de code suivant montre comment utiliser `EnableMetricsCollection`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [EnableMetricsCollection](#) à la section Référence des AWS SDK for Java 2.x API.

SetDesiredCapacity

L'exemple de code suivant montre comment utiliser `SetDesiredCapacity`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");


    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SetDesiredCapacity](#) à la section Référence des AWS SDK for Java 2.x API.

TerminateInstanceInAutoScalingGroup

L'exemple de code suivant montre comment utiliser `TerminateInstanceInAutoScalingGroup`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
        TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstanceInAutoScalingGroup](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateAutoScalingGroup

L'exemple de code suivant montre comment utiliser `UpdateAutoScalingGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
        LaunchTemplateSpecification.builder()
```

```
        .launchTemplateName(launchTemplateName)
        .build();

    UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
        .maxSize(3)
        .autoScalingGroupName(groupName)
        .launchTemplate(templateSpecification)
        .build();

    autoScalingClient.updateAutoScalingGroup(groupRequest);
    DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateAutoScalingGroup](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Créer et gérer un service résilient

L'exemple de code suivant montre comment créer un service Web à charge équilibrée qui renvoie des recommandations de livres, de films et de chansons. L'exemple montre comment le service répond aux défaillances et comment le restructurer pour accroître la résilience en cas de défaillance.

- Utilisez un groupe Amazon EC2 Auto Scaling pour créer des instances Amazon Elastic Compute Cloud (Amazon EC2) sur la base d'un modèle de lancement et pour maintenir le nombre d'instances dans une plage spécifiée.
- Gérez et distribuez les requêtes HTTP avec Elastic Load Balancing.
- Surveillez l'état des instances d'un groupe Auto Scaling et transférez les demandes uniquement aux instances saines.
- Exécutez un serveur Web Python sur chaque instance EC2 pour gérer les requêtes HTTP. Le serveur Web répond par des recommandations et des surveillances de l'état.
- Simulez un service de recommandation avec une table Amazon DynamoDB.
- Contrôlez la réponse du serveur Web aux demandes et aux contrôles de santé en mettant à jour AWS Systems Manager les paramètres.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
    public static final String policyFile = "C:\\AWS\\resworkflow\\  
\\instance_policy.json"; // Modify file location.  
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\  
\\ssm_only_policy.json"; // Modify file location.  
    public static final String failureResponse = "doc-example-resilient-  
architecture-failure-response";  
    public static final String healthCheck = "doc-example-resilient-architecture-  
health-check";  
    public static final String templateName = "doc-example-resilience-template";  
    public static final String roleName = "doc-example-resilience-role";  
}
```



```
public static final String policyName = "doc-example-resilience-pol";
public static final String profileName = "doc-example-resilience-prof";

public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

public static final String targetGroupName = "doc-example-resilience-tg";
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println(""
```

This concludes the demo of how to build and manage a resilient service.

To keep things tidy and to avoid unwanted charges on your account, we can clean up all AWS resources

that were created for this demo.

```
""");
```

```
System.out.println("\n Do you want to delete the resources (y/n)? ");
```

```
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input
```

```
if (userInput.equals("y")) {
```

```
    // Delete resources here
```

```
    deleteResources(loadBalancer, autoScaler, database);
```

```
    System.out.println("Resources deleted.");
```

```
} else {
```

```
    System.out.println("""
```

```
        Okay, we'll leave the resources intact.
```

```
        Don't forget to delete them when you're done with them or you
```

might incur unexpected charges.

```
        """);
```

```
}
```

```
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("The example has completed. ");
```

```
System.out.println("\n Thanks for watching!");
```

```
System.out.println(DASHES);
```

```
}
```

```
// Deletes the AWS resources used in this example.
```

```
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler autoScaler, Database database)
```

```
    throws IOException, InterruptedException {
```

```
    loadBalancer.deleteLoadBalancer(lbName);
```

```
    System.out.println("*** Wait 30 secs for resource to be deleted");
```

```
    TimeUnit.SECONDS.sleep(30);
```

```
    loadBalancer.deleteTargetGroup(targetGroupName);
```

```
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
```

```
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
```

```
    autoScaler.deleteTemplate(templateName);
```

```
    database.deleteTable(tableName);
```

```
}
```

```

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.
    """);

```

```
        The template also defines an IAM policy that each instance uses to
assume a role that grants
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);
```

```
        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);
```

```
        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
```

```

        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
            // Execute the request and get the response
            HttpResponse response = httpClient.execute(httpGet);

            // Read the response content.
            String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

            // Print the public IP address.
            System.out.println("Public IP Address: " + ipAddress);
            GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
            if (!groupInfo.isPortOpen()) {
                System.out.println("""
                    For this example to work, the default security group for
your default VPC must
                    allow access from this computer. You can either add it
automatically from this

```

```

        example or add it yourself using the AWS Management
Console.
        """);

        System.out.println(
            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
        System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();

```

```
System.out.println("Read the ssm_only_policy.json file");
String ssmOnlyPolicy = readFileAsString(ssmJSON);

System.out.println("Resetting parameters to starting values for demo.");
paramHelper.reset();

System.out.println(
    """
        This part of the demonstration shows how to toggle
different parts of the system
        to create situations where the web service fails, and shows
how using a resilient
        architecture can keep the web service running in spite of
these failures.

        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
    """);
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    """
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        Instead of failing when the recommendation service fails,
the web service can return a static response.
    """);
```

```
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
    paramHelper.put(paramHelper.failureResponse, "static");

    System.out.println("""
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
    demoChoices(loadBalancer);

    System.out.println("Let's reinstate the recommendation service.");
    paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

    System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    AutoScaler autoScaler = new AutoScaler();

    // Create a new instance profile based on badCredsProfileName.
    templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
    String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
    System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

    String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
    System.out.println("The association Id value is " + profileAssociationId);
    System.out.println("Replacing the profile for instance " + badInstanceId
        + " with a profile that contains bad credentials");
    autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

    System.out.println(
        """"
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
```



```

                                depending on which instance is selected by the load
balancer.
                                """);

    demoChoices(loadBalancer);

    System.out.println("""
        Let's implement a deep health check. For this demo, a deep health
check tests whether
        the web service can access the DynamoDB table that it depends on for
recommendations. Note that
        the deep health check is only for ELB routing and not for Auto
Scaling instance health.
        This kind of deep health check is not recommended for Auto Scaling
instance health, because it
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
        """);

    System.out.println("""
        By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
        """);

    paramHelper.put(paramHelper.healthCheck, "deep");

    System.out.println("""
        Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
        """);

    demoChoices(loadBalancer);

    System.out.println(
        ""

        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
```

```

        instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {

```

```
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClientBuilder.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();

                // Print the formatted JSON response.
                System.out.println("Full Response:\n");
                System.out.println(jsonResponse.toString());

                // Close the HTTP client.
                httpClient.close();
            }
        }
    }
```

```

        case 1 -> {
            System.out.println("\nChecking the health of load balancer
targets:\n");
            List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
            for (TargetHealthDescription target : health) {
                System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
            }
            System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
        }
        case 2 -> {
            System.out.println("\nOkay, let's move on.");
            System.out.println("-".repeat(88));
            return; // Exit the method when choice is 2
        }
        default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Créez une classe qui englobe les actions Auto Scaling et Amazon EC2.

```
public class AutoScaler {
```

```
private static Ec2Client ec2Client;
private static AutoScalingClient autoScalingClient;
private static IamClient iamClient;

private static SsmClient ssmClient;

private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
```

```

    * Terminates and instances in an EC2 Auto Scaling group. After an instance is
    * terminated, it can no longer be accessed.
    */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
        TerminateInstanceInAutoScalingGroupRequest
            .builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
    newInstanceProfileName, String profileAssociationId)
        throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
        iamInstanceProfile =
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
            .builder()
            .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
            // name.
            .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
        ReplaceIamInstanceProfileAssociationRequest
            .builder()
            .iamInstanceProfile(iamInstanceProfile)
            .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
            .build();
    }

```

```
try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}
System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.getInstanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
```

```

        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
            Collections.singletonList("cd / && sudo python3 server.py
80")))
        .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.getInstanceProfile().getInstanceProfileName();
            System.out.println(name);
        }
    }

```



```
        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(profileName)
    .roleName(roleName)
    .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(profileName)
    .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
    .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(attachedPolicy.policyArn())
    .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();
```

```

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
        .filters(filter, filter1)
        .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
        .describeSecurityGroups(securityGroupsRequest);

        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }

                    if (!ipPermission.prefixListIds().isEmpty()) {
                        System.out.println("Prefix lList is applicable");
                        portIsOpen = true;
                    }

                    if (!portIsOpen) {
                        System.out
                            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                    } else {
                        break;
                    }
                }
            }
        }
    }
}

```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }

    groupInfo.setPortOpen(portIsOpen);
    return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
        .builder()
        .build();
```

```
        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
        .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();
```

```
        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        Filter azFilter = Filter.builder()
            .name("availability-zone")
            .values(availabilityZones)
            .build();

        Filter defaultForAZ = Filter.builder()
            .name("default-for-az")
            .values("true")
            .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
            .filters(vpcFilter, azFilter, defaultForAZ)
            .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();
```

```
DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
List<String> instanceIds = autoScalingGroup.instances().stream()
    .map(instance -> instance.instanceId())
    .collect(Collectors.toList());

String[] instanceIdArray = instanceIds.toArray(new String[0]);
for (String instanceId : instanceIdArray) {
    System.out.println("Instance ID: " + instanceId);
    return instanceId;
}
return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

```

```

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
    ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
        .listEntitiesForPolicy(listEntitiesRequest);
    if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
        || !listEntitiesResponse.policyRoles().isEmpty()) {
        // Detach the policy from any entities it is attached to.
        DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(policy.arn())
            .roleName(roleName) // Specify the name of the IAM role
            .build();

        getIAMClient().detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
        .policyArn(policy.arn())
        .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);

```



```

        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .roleName(roleName) // Remove the extra dot here
                .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(InstanceProfile)
            .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Créez une classe qui englobe les actions Elastic Load Balancing.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.

```

```
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
```

```
        .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
            }
        }
    }
}
```

```
        System.out.println("Got connection error from load balancer
endpoint, retrying...");
        TimeUnit.SECONDS.sleep(15);
    }
}

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
```

```
* Creates an Elastic Load Balancing load balancer that uses the specified
* subnets
* and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
```

```

        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
}

```

Créez une classe qui utilise DynamoDB pour simuler un service de recommandation.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }
}

```

```
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()

```

```

        .attributeName("ItemId")
        .attributeType(ScalarAttributeType.N)
        .build())
    .keySchema(
        KeySchemaElement.builder()
            .attributeName("MediaType")
            .keyType(KeyType.HASH)
            .build(),
        KeySchemaElement.builder()
            .attributeName("ItemId")
            .keyType(KeyType.RANGE)
            .build())
    .provisionedThroughput(
        ProvisionedThroughput.builder()
            .readCapacityUnits(5L)
            .writeCapacityUnits(5L)
            .build())
    .build();

    getDynamoDbClient().createTable(createTableRequest);
    System.out.println("Creating table " + tableName + "...");

    // Wait until the Amazon DynamoDB table is created.
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

    // Add records to the table.
    populateTable(fileName, tableName);
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.

```



```

public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}

```

Créez une classe qui englobe les actions de Systems Manager.

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
    }
}

```

```
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)

- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Gérer les groupes et les instances

L'exemple de code suivant illustre comment :

- Créez un groupe Amazon EC2 Auto Scaling avec un modèle de lancement et des zones de disponibilité, et obtenez des informations sur les instances en cours d'exécution.
- Activez la collecte CloudWatch de métriques Amazon.
- Mettez à jour la capacité souhaitée du groupe et attendez qu'une instance démarre.
- Mettez fin à une instance du groupe.
- Répertoriez les activités de dimensionnement qui se produisent en réponse aux demandes des utilisateurs et aux modifications de capacité.
- Obtenez des statistiques pour CloudWatch les métriques, puis nettoyez les ressources.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
    }
}
```

```
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String groupName = args[0];
    String launchTemplateName = args[1];
    String vpcZoneId = args[2];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an Auto Scaling group named " + groupName);
    createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
    System.out.println(
        "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
    Thread.sleep(60000);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get Auto Scale group Id value");
    String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
    if (instanceId.compareTo("") == 0) {
        System.out.println("Error - no instance Id value");
        System.exit(1);
    } else {
        System.out.println("The instance Id value is " + instanceId);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
            "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}

public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
String launchTemplateName,
String vpcZoneId) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
    }
}
```



```
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
```

```
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .maxRecords(10)
                .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                }
            }
        }
    }
}
```

```
        System.out.println("The lifecycle state is " +
instance.lifecycleState());
    }
}

    return instanceId;
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
```

```
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is " +
response.numberOfWorkingAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
```

```
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
```

```
        System.out.println("You successfully deleted " + groupName);
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Exemples d'Amazon Bedrock utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Bedrock.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

GetFoundationModel

L'exemple de code suivant montre comment utiliser `GetFoundationModel`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez des informations sur un modèle de base à l'aide du client synchrone Amazon Bedrock.

```
/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
    try {
        GetFoundationModelResponse response = bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)
        );

        FoundationModelDetails model = response.modelDetails();

        System.out.println(" Model ID:                " + model.modelId());
        System.out.println(" Model ARN:                " +
model.modelArn());
    }
}
```

```

        System.out.println(" Model Name: " +
model.modelName());
        System.out.println(" Provider Name: " +
model.providerName());
        System.out.println(" Lifecycle status: " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities: " +
model.inputModalities());
        System.out.println(" Output modalities: " +
model.outputModalities());
        System.out.println(" Supported customizations: " +
model.customizationsSupported());
        System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

        return model;

    } catch (ValidationException e) {
        throw new IllegalArgumentException(e.getMessage());
    } catch (SdkException e) {
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

Obtenez des informations sur un modèle de base à l'aide du client asynchrone Amazon Bedrock.

```

/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
    try {
        CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)

```



```

        );

        FoundationModelDetails model = future.get().modelDetails();

        System.out.println(" Model ID:                " + model.modelId());
        System.out.println(" Model ARN:                " +
model.modelArn());
        System.out.println(" Model Name:                " +
model.modelName());
        System.out.println(" Provider Name:            " +
model.providerName());
        System.out.println(" Lifecycle status:        " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities:        " +
model.inputModalities());
        System.out.println(" Output modalities:       " +
model.outputModalities());
        System.out.println(" Supported customizations: " +
model.customizationsSupported());
        System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

        return model;

    } catch (ExecutionException e) {
        if (e.getMessage().contains("ValidationException")) {
            throw new IllegalArgumentException(e.getMessage());
        } else {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [GetFoundationModel](#) à la section Référence des AWS SDK for Java 2.x API.

ListFoundationModels

L'exemple de code suivant montre comment utiliser `ListFoundationModels`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertoriez les modèles de fondation Amazon Bedrock disponibles à l'aide du client synchrone Amazon Bedrock.

```
/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary> listFoundationModels(BedrockClient
bedrockClient) {

    try {
        ListFoundationModelsResponse response =
bedrockClient.listFoundationModels(r -> {});

        List<FoundationModelSummary> models = response.modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }
    }
}
```

```

        return models;

    } catch (SdkClientException e) {
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

Répertoriez les modèles de fondation Amazon Bedrock disponibles à l'aide du client asynchrone Amazon Bedrock.

```

/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
    try {
        CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

        List<FoundationModelSummary> models = future.get().modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }

        return models;

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }
}

```

```
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListFoundationModels](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'exécution Amazon Bedrock utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Bedrock Runtime.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [AI21 Labs Jurassic 2](#)
- [Amazon Titan Image Generator](#)
- [Texte Amazon Titan](#)
- [Intégrations de texte Amazon Titan](#)
- [Anthropic Claude](#)
- [Cohere Command](#)
- [Méta lama](#)
- [IA Mistral](#)
- [Scénarios](#)

- [Stable Diffusion](#)

AI21 Labs Jurassic 2

Converse

L'exemple de code suivant montre comment envoyer un message texte à AI21 Labs Jurassic-2 à l'aide de l'API Converse de Bedrock.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un message texte à AI21 Labs Jurassic-2 à l'aide de l'API Converse de Bedrock.

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Jurassic-2 Mid.
var modelId = "ai21.j2-mid-v1";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

try {
    // Send the message with a basic inference configuration.
    ConverseResponse response = client.converse(request -> request
        .modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    converse();
}
}
```

Envoyez un message texte à AI21 Labs Jurassic-2 en utilisant l'API Converse de Bedrock avec le client Java asynchrone.

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
```

```
        .temperature(0.5F)
        .topP(0.9F))
    );

    // Prepare a future object to handle the asynchronous response.
    CompletableFuture<String> future = new CompletableFuture<>();

    // Handle the response or error using the future object.
    request.whenComplete((response, error) -> {
        if (error == null) {
            // Extract the generated text from Bedrock's response object.
            String responseText =
response.output().message().content().get(0).text();
            future.complete(responseText);
        } else {
            future.completeExceptionally(error);
        }
    });

    try {
        // Wait for the future object to complete and retrieve the generated
text.
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;
    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- Pour plus de détails sur l'API, consultez [Converse dans le guide](#) de référence des AWS SDK for Java 2.x API.

InvokeModel

L'exemple de code suivant montre comment envoyer un message texte à AI21 Labs Jurassic-2 à l'aide de l'API Invoke Model.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte.

```
// Use the native inference API to send a text message to AI21 Labs Jurassic-2.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
```

```
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
jurassic2.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/completions/0/data/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

Amazon Titan Image Generator

InvokeModel

L'exemple de code suivant montre comment invoquer Amazon Titan Image sur Amazon Bedrock pour générer une image.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez une image avec le générateur d'images Amazon Titan.

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Titan Image G1.
var modelId = "amazon.titan-image-generator-v1";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
var nativeRequestTemplate = """
    {
        "taskType": "TEXT_IMAGE",
        "textToImageParams": { "text": "{{prompt}}" },
        "imageGenerationConfig": { "seed": {{seed}} }
    }""";

// Define the prompt for the image generation.
var prompt = "A stylized picture of a cute old steampunk robot";

// Get a random 31-bit seed for the image generation (max. 2,147,483,647).
var seed = new BigInteger(31, new SecureRandom());

// Embed the prompt and seed in the model's native request payload.
var nativeRequest = nativeRequestTemplate
    .replace("{{prompt}}", prompt)
    .replace("{{seed}}", seed.toString());

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated image data from the model's response.
    var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

    return base64ImageData;

} catch (SdkClientException e) {
```

```
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    System.out.println("Generating image. This may take a few seconds...");

    String base64ImageData = invokeModel();

    displayImage(base64ImageData);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

Texte Amazon Titan

Converse

L'exemple de code suivant montre comment envoyer un message texte à Amazon Titan Text à l'aide de l'API Converse de Bedrock.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un SMS à Amazon Titan Text à l'aide de l'API Converse de Bedrock.

```
// Use the Converse API to send a text message to Amazon Titan Text.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);
        }
    }
}
```

```
        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

Envoyez un message texte à Amazon Titan Text à l'aide de l'API Converse de Bedrock avec le client Java asynchrone.

```
// Use the Converse API to send a text message to Amazon Titan Text
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Titan Text Premier.
var modelId = "amazon.titan-text-premier-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);
}
```



```
        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- Pour plus de détails sur l'API, consultez [Converse dans le guide](#) de référence des AWS SDK for Java 2.x API.

ConverseStream

L'exemple de code suivant montre comment envoyer un message texte à Amazon Titan Text à l'aide de l'API Converse de Bedrock et comment traiter le flux de réponses en temps réel.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un SMS à Amazon Titan Text à l'aide de l'API Converse de Bedrock et traitez le flux de réponses en temps réel.

```
// Use the Converse API to send a text message to Amazon Titan Text
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
```

```
import
software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
            ).build();

        try {
```

```
// Send the message with a basic inference configuration and attach the
handler.
client.converseStream(request -> request
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F)
    ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ConverseStream](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModel

L'exemple de code suivant montre comment envoyer un message texte à Amazon Titan Text à l'aide de l'API Invoke Model.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte.

```
// Use the native inference API to send a text message to Amazon Titan Text.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
```

```
        var text = new JSONPointer("/results/0/outputText").queryFrom(responseBody).toString();
        System.out.println(text);

        return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModelWithResponseStream

L'exemple de code suivant montre comment envoyer un message texte aux modèles Amazon Titan Text, à l'aide de l'API Invoke Model, et comment imprimer le flux de réponses.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte et traiter le flux de réponses en temps réel.

```
// Use the native inference API to send a text message to Amazon Titan Text
// and print the response stream.
```

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);
```

```
// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputText").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

    // Return the complete response text.
    return completeResponseTextBuffer.toString();

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModelWithResponseStream](#) à la section Référence des AWS SDK for Java 2.x API.

Intégrations de texte Amazon Titan

InvokeModel

L'exemple de code suivant illustre comment :

- Commencez à créer votre première intégration.
- Créez des intégrations configurant le nombre de dimensions et la normalisation (V2 uniquement).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez votre première intégration avec Titan Text Embeddings V2.

```
// Generate and print an embedding with Amazon Titan Text Embeddings.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```



```
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Titan Text Embeddings V2.
var modelId = "amazon.titan-embed-text-v2:0";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-embed-text.html
var nativeRequestTemplate = "{ \"inputText\": \"{{inputText}}\" }";

// The text to convert into an embedding.
var inputText = "Please recommend books with a theme similar to the movie
'Inception.'";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{inputText}}",
inputText);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
embedding").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;
} catch (SdkClientException e) {
```

```

        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}

```

Appelez Titan Text Embeddings V2 pour configurer le nombre de dimensions et la normalisation.

```

/**
 * Invoke Amazon Titan Text Embeddings V2 with additional inference parameters.
 *
 * @param inputText - The text to convert to an embedding.
 * @param dimensions - The number of dimensions the output embeddings should
have.
 *
 * Values accepted by the model: 256, 512, 1024.
 * @param normalize - A flag indicating whether or not to normalize the output
embeddings.
 * @return The {@link JSONObject} representing the model's response.
 */
public static JSONObject invokeModel(String inputText, int dimensions, boolean
normalize) {

    // Create a Bedrock Runtime client in the AWS Region of your choice.
    var client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .build();

    // Set the model ID, e.g., Titan Embed Text v2.0.
    var modelId = "amazon.titan-embed-text-v2:0";

    // Create the request for the model.
    var nativeRequest = ""
        {
            "inputText": "%s",
            "dimensions": %d,
            "normalize": %b

```

```
    }
    """.formatted(inputText, dimensions, normalize);

    // Encode and send the request.
    var response = client.invokeModel(request -> {
        request.body(SdkBytes.fromUtf8String(nativeRequest));
        request.modelId(modelId);
    });

    // Decode the model's response.
    var modelResponse = new JSONObject(response.body().asUtf8String());

    // Extract and print the generated embedding and the input text token count.
    var embedding = modelResponse.getJSONArray("embedding");
    var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
    System.out.println("Embedding: " + embedding);
    System.out.println("\nInput token count: " + inputTokenCount);

    // Return the model's native response.
    return modelResponse;
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

Anthropic Claude

Converse

L'exemple de code suivant montre comment envoyer un message texte à Anthropic Claude à l'aide de l'API Converse de Bedrock.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un SMS à Anthropic Claude à l'aide de l'API Converse de Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
```

```

        .temperature(0.5F)
        .topP(0.9F)));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converse();
}
}

```

Envoyez un SMS à Anthropic Claude en utilisant l'API Converse de Bedrock avec le client Java asynchrone.

```

// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.

```

```
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});
```

```
        try {
            // Wait for the future object to complete and retrieve the generated
            text.

            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- Pour plus de détails sur l'API, consultez [Converse dans le guide](#) de référence des AWS SDK for Java 2.x API.

ConverseStream

L'exemple de code suivant montre comment envoyer un message texte à Anthropic Claude à l'aide de l'API Converse de Bedrock et comment traiter le flux de réponses en temps réel.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un SMS à Anthropic Claude à l'aide de l'API Converse de Bedrock et traitez le flux de réponses en temps réel.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
```



```
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
        ).build();

    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request.modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ConverseStream](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModel

L'exemple de code suivant montre comment envoyer un message texte à Anthropic Claude à l'aide de l'API Invoke Model.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte.

```
// Use the native inference API to send a text message to Anthropic Claude.
```

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        anthropic-claude-messages.html
        var nativeRequestTemplate = ""
            {
                "anthropic_version": "bedrock-2023-05-31",
                "max_tokens": 512,
                "temperature": 0.5,
                "messages": [{
                    "role": "user",
                    "content": "{{prompt}}"
                }]
            }"";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
```

```
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/content/0/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModelWithResponseStream

L'exemple de code suivant montre comment envoyer un message texte aux modèles Anthropic Claude, à l'aide de l'API Invoke Model, et comment imprimer le flux de réponses.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte et traiter le flux de réponses en temps réel.

```
// Use the native inference API to send a text message to Anthropic Claude
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.Objects;
import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
```

```
        .build();

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
var nativeRequestTemplate = """
    {
      "anthropic_version": "bedrock-2023-05-31",
      "max_tokens": 512,
      "temperature": 0.5,
      "messages": [{
        "role": "user",
        "content": "{{prompt}}"
      }]
    }""";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        var response = new JSONObject(chunk.bytes().asUtf8String());
```

```
        // Extract and print the text from the content blocks.
        if (Objects.equals(response.getString("type"),
"content_block_delta")) {
            var text = new JSONPointer("/delta/
text").queryFrom(response);
            System.out.print(text);

            // Append the text to the response text buffer.
            completeResponseTextBuffer.append(text);
        }
    }).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

    public static void main(String[] args) throws ExecutionException,
InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModelWithResponseStream](#) à la section Référence des AWS SDK for Java 2.x API.

Cohere Command

Converse : Tous les modèles

L'exemple de code suivant montre comment envoyer un message texte à Cohere Command, à l'aide de l'API Converse de Bedrock.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un message texte à Cohere Command, en utilisant l'API Converse de Bedrock.

```
// Use the Converse API to send a text message to Cohere Command.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";
```

```
    // Create the input text and embed it in a message object with the user
    role.
    var inputText = "Describe the purpose of a 'hello world' program in one
    line.";
    var message = Message.builder()
        .content(ContentBlock.fromText(inputText))
        .role(ConversationRole.USER)
        .build();

    try {
        // Send the message with a basic inference configuration.
        ConverseResponse response = client.converse(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)));

        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

Envoyez un message texte à Cohere Command, en utilisant l'API Converse de Bedrock avec le client Java asynchrone.

```
// Use the Converse API to send a text message to Cohere Command
```



```
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F))
```

```
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converseAsync();
}
}
```

- Pour plus de détails sur l'API, consultez [Converse dans le guide](#) de référence des AWS SDK for Java 2.x API.

ConverseStream: Tous les modèles

L'exemple de code suivant montre comment envoyer un message texte à Cohere Command à l'aide de l'API Converse de Bedrock et comment traiter le flux de réponse en temps réel.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un message texte à Cohere Command à l'aide de l'API Converse de Bedrock et traitez le flux de réponse en temps réel.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Create a handler to extract and print the response text in real-time.
var responseStreamHandler = ConverseStreamResponseHandler.builder()
    .subscriber(ConverseStreamResponseHandler.Visitor.builder()
        .onContentBlockDelta(chunk -> {
            String responseText = chunk.delta().text();
            System.out.print(responseText);
        }).build()
    ).onError(err ->
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
    ).build();

try {
    // Send the message with a basic inference configuration and attach the
handler.
    client.converseStream(request -> request.modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)
        ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ConverseStream](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModel: Command R et R+

L'exemple de code suivant montre comment envoyer un message texte à Cohere Command R et R+ à l'aide de l'API Invoke Model.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte.

```
// Use the native inference API to send a text message to Cohere Command R.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_R_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";
```

```
// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}


public static void main(String[] args) {
    invokeModel();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModel: lampe de commande et de commande

L'exemple de code suivant montre comment envoyer un message texte à Cohere Command à l'aide de l'API Invoke Model.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte.

```
// Use the native inference API to send a text message to Cohere Command.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command Light.
        var modelId = "cohere.command-light-text-v14";
```

```
// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/generations/0/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    invokeModel();
}
}
```


- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModelWithResponseStream: Command R et R+

L'exemple de code suivant montre comment envoyer un message texte à Cohere Command, à l'aide de l'API Invoke Model avec un flux de réponses.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte et traiter le flux de réponses en temps réel.

```
// Use the native inference API to send a text message to Cohere Command R
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Command_R_InvokeModelWithResponseStream {
```

```
public static String invokeModelWithResponseStream() throws ExecutionException,
InterruptedException {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    // Replace the DefaultCredentialsProvider with your preferred credentials
    provider.
    var client = BedrockRuntimeAsyncClient.builder()
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

    // Set the model ID, e.g., Command R.
    var modelId = "cohere.command-r-v1:0";

    // The InvokeModelWithResponseStream API uses the model's native payload.
    // Learn more about the available inference parameters and response fields
    at:
    // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    cohere-command-r-plus.html
    var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

    // Define the prompt for the model.
    var prompt = "Describe the purpose of a 'hello world' program in one line.";

    // Embed the prompt in the model's native request payload.
    String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

    // Create a request with the model ID and the model's native request
    payload.
    var request = InvokeModelWithResponseStreamRequest.builder()
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
        .build();

    // Prepare a buffer to accumulate the generated response text.
    var completeResponseTextBuffer = new StringBuilder();

    // Prepare a handler to extract, accumulate, and print the response text in
    real-time.
    var responseStreamHandler =
    InvokeModelWithResponseStreamResponseHandler.builder()
        .subscriber(Visitor.builder().onChunk(chunk -> {
            // Extract and print the text from the model's native response.
            var response = new JSONObject(chunk.bytes().asUtf8String());
```

```
        var text = new JSONPointer("/text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModelWithResponseStream: lampe de commande et de commande

L'exemple de code suivant montre comment envoyer un message texte à Cohere Command, à l'aide de l'API Invoke Model avec un flux de réponses.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte et traiter le flux de réponses en temps réel.

```
// Use the native inference API to send a text message to Cohere Command
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Command_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Command Light.
var modelId = "cohere.command-light-text-v14";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generations/0/
text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
```

```
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

Méta lama

Tous les modèles : Converse API

L'exemple de code suivant montre comment envoyer un message texte à Meta Llama à l'aide de l'API Converse de Bedrock.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un SMS à Meta Llama à l'aide de l'API Converse de Bedrock.

```
// Use the Converse API to send a text message to Meta Llama.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));
        }
    }
}
```

```
        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

Envoyez un message texte à Meta Llama en utilisant l'API Converse de Bedrock avec le client Java asynchrone.

```
// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
```



```
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
```

```
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- Pour plus de détails sur l'API, consultez [Converse dans le guide](#) de référence des AWS SDK for Java 2.x API.

ConverseStream: Tous les modèles

L'exemple de code suivant montre comment envoyer un message texte à Meta Llama à l'aide de l'API Converse de Bedrock et comment traiter le flux de réponses en temps réel.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un message texte à Meta Llama à l'aide de l'API Converse de Bedrock et traitez le flux de réponses en temps réel.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
            ).build();
```

```
    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();


    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ConverseStream](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModel: Lama 2

L'exemple de code suivant montre comment envoyer un message texte à Meta Llama 2 à l'aide de l'API Invoke Model.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte.

```
// Use the native inference API to send a text message to Meta Llama 2.

import org.json.JSONObject;
```

```
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama2_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 2 Chat 13B.
        var modelId = "meta.llama2-13b-chat-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\n\"}";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 2's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\n".replace("{{prompt}}",
        prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest)))
```

```
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModel: Lama 3

L'exemple de code suivant montre comment envoyer un message texte à Meta Llama 3 à l'aide de l'API Invoke Model.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte.

```
// Use the native inference API to send a text message to Meta Llama 3.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama3_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 3's instruction format.
        var instruction = (
            "<|begin_of_text|>\n" +
            "<|start_header_id|>user<|end_header_id|>\n" +
            "{{prompt}} <|eot_id|>\n" +
            "<|start_header_id|>assistant<|end_header_id|>\n"
        ).replace("{{prompt}}", prompt);
```

```
// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModelWithResponseStream: Lama 2

L'exemple de code suivant montre comment envoyer un message texte à Meta Llama 2, à l'aide de l'API Invoke Model, et comment imprimer le flux de réponse.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte et traiter le flux de réponses en temps réel.

```
// Use the native inference API to send a text message to Meta Llama 2
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama2_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Llama 2 Chat 13B.
var modelId = "meta.llama2-13b-chat-v1";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 2's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    })
    )
    .build();
```

```
        }).build()).build());

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModelWithResponseStream](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModelWithResponseStream: Lama 3

L'exemple de code suivant montre comment envoyer un message texte à Meta Llama 3, à l'aide de l'API Invoke Model, et comment imprimer le flux de réponse.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte et traiter le flux de réponses en temps réel.

```
// Use the native inference API to send a text message to Meta Llama 3
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama3_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
```

```
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 3's instruction format.
var instruction = (
    "<|begin_of_text|>\n" +
    "<|start_header_id|>user<|end_header_id|>\n" +
    "{{prompt}} <|eot_id|>\n" +
    "<|start_header_id|>assistant<|end_header_id|>\n"
).replace("{{prompt}}", prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
```

```
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModelWithResponseStream](#) à la section Référence des AWS SDK for Java 2.x API.

IA Mistral

Converse

L'exemple de code suivant montre comment envoyer un message texte à Mistral à l'aide de l'API Converse de Bedrock.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un SMS à Mistral en utilisant l'API Converse de Bedrock.

```
// Use the Converse API to send a text message to Mistral.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));
        }
    }
}
```

```
        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }

}

public static void main(String[] args) {
    converse();
}
}
```

Envoyez un SMS à Mistral en utilisant l'API Converse de Bedrock avec le client Java asynchrone.

```
// Use the Converse API to send a text message to Mistral
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
```



```
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
```

```
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- Pour plus de détails sur l'API, consultez [Converse dans le guide](#) de référence des AWS SDK for Java 2.x API.

ConverseStream

L'exemple de code suivant montre comment envoyer un message texte à Mistral à l'aide de l'API Converse de Bedrock et comment traiter le flux de réponses en temps réel.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyez un SMS à Mistral à l'aide de l'API Converse de Bedrock et traitez le flux de réponses en temps réel.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
                err.getMessage())
            ).build();
```

```
    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request.modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ConverseStream](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModel

L'exemple de code suivant montre comment envoyer un message texte aux modèles Mistral à l'aide de l'API Invoke Model.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte.

```
// Use the native inference API to send a text message to Mistral.

import org.json.JSONObject;
import org.json.JSONPointer;
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Mistral's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
        prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
```

```
        );

        // Decode the response body.
        var responseBody = new JSONObject(response.body().asUtf8String());

        // Retrieve the generated text from the model's response.
        var text = new JSONPointer("/outputs/0/
text").queryFrom(responseBody).toString();
        System.out.println(text);

        return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

InvokeModelWithResponseStream

L'exemple de code suivant montre comment envoyer un message texte aux modèles Mistral AI, à l'aide de l'API Invoke Model, et comment imprimer le flux de réponses.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez l'API Invoke Model pour envoyer un message texte et traiter le flux de réponses en temps réel.

```
// Use the native inference API to send a text message to Mistral
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        mistral-text-completion.html
```

```
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Mistral's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputs/0/
text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

    // Return the complete response text.
```



```
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModelWithResponseStream](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Créez une application de terrain de jeu pour interagir avec les modèles de la fondation Amazon Bedrock

L'exemple de code suivant montre comment créer des terrains de jeu pour interagir avec les modèles de fondation Amazon Bedrock selon différentes modalités.

SDK pour Java 2.x

Le Java Foundation Model (FM) Playground est un exemple d'application Spring Boot qui montre comment utiliser Amazon Bedrock avec Java. Cet exemple montre comment les développeurs Java peuvent utiliser Amazon Bedrock pour créer des applications génératives basées sur l'IA. Vous pouvez tester et interagir avec les modèles de fondation Amazon Bedrock en utilisant les trois terrains de jeu suivants :

- Un terrain de jeu pour les textes.
- Un terrain de jeu pour le chat.
- Un terrain de jeu pour l'image.

L'exemple répertorie et affiche également les modèles de base auxquels vous avez accès, ainsi que leurs caractéristiques. Pour le code source et les instructions de déploiement, consultez le projet dans [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Bedrock Runtime

Stable Diffusion

InvokeModel

L'exemple de code suivant montre comment invoquer Stability.ai Stable Diffusion XL sur Amazon Bedrock pour générer une image.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez une image avec Stable Diffusion.

```
// Create an image with Stable Diffusion.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {
```

```
public static String invokeModel() {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    // Replace the DefaultCredentialsProvider with your preferred credentials
    provider.
    var client = BedrockRuntimeClient.builder()
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

    // Set the model ID, e.g., Stable Diffusion XL v1.
    var modelId = "stability.stable-diffusion-xl-v1";

    // The InvokeModel API uses the model's native payload.
    // Learn more about the available inference parameters and response fields
    at:
    // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    diffusion-1-0-text-image.html
    var nativeRequestTemplate = ""
        {
            "text_prompts": [{ "text": "{{prompt}}" }],
            "style_preset": "{{style}}",
            "seed": {{seed}}
        }"";

    // Define the prompt for the image generation.
    var prompt = "A stylized picture of a cute old steampunk robot";

    // Get a random 32-bit seed for the image generation (max. 4,294,967,295).
    var seed = new BigInteger(31, new SecureRandom());

    // Choose a style preset.
    var style = "cinematic";

    // Embed the prompt, seed, and style in the model's native request payload.
    String nativeRequest = nativeRequestTemplate
        .replace("{{prompt}}", prompt)
        .replace("{{seed}}", seed.toString())
        .replace("{{style}}", style);

    try {
        // Encode and send the request to the Bedrock Runtime.
        var response = client.invokeModel(request -> request
            .body(SdkBytes.fromUtf8String(nativeRequest))
```

```
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated image data from the model's response.
    var base64ImageData = new JSONPointer("/artifacts/0/base64")
        .queryFrom(responseBody)
        .toString();

    return base64ImageData;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
    System.out.println("Generating image. This may take a few seconds...");

    String base64ImageData = invokeModel();

    displayImage(base64ImageData);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [InvokeModel](#) à la section Référence des AWS SDK for Java 2.x API.

CloudFront exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with CloudFront.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateDistribution

L'exemple de code suivant montre comment utiliser `CreateDistribution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

L'exemple suivant utilise un bucket Amazon Simple Storage Service (Amazon S3) comme origine de contenu.

Après avoir créé la distribution, le code crée un [CloudFrontWaiter](#) pour attendre que la distribution soit déployée avant de renvoyer la distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
```

```
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

    private static final Logger logger =
        LoggerFactory.getLogger(CreateDistribution.class);

    public static Distribution createDistribution(CloudFrontClient
        cloudFrontClient, S3Client s3Client,
            final String bucketName, final String keyGroupId, final
            String originAccessControlId) {

        final String region = s3Client.headBucket(b ->
            b.bucket(bucketName)).sdkHttpResponse().headers()
            .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
            ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for
        the originId.

        // The service API requires some deprecated methods, such as
        // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
        CreateDistributionResponse createDistResponse =
            cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                    .origins(b2 -> b2
                        .quantity(1)
                        .items(b3 -> b3

                    .domainName(originDomain)

                    .id(originId)

                    .s3OriginConfig(builder4 -> builder4

                        .originAccessIdentity(

                            "")))
```

```

.originAccessControlId(
    originAccessControlId)))
    .defaultCacheBehavior(b2 -> b2
    .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
    .targetOriginId(originId)
    .minTTL(200L)
    .forwardedValues(b5
-> b5
    .cookies(cp -> cp
    .forward(ItemSelection.NONE))
    .queryString(true))
    .trustedKeyGroups(b3
-> b3
    .quantity(1)
    .items(keyGroupId)
    .enabled(true))
    .allowedMethods(b4 -
> b4
    .quantity(2)
    .items(Method.HEAD, Method.GET)
    .cachedMethods(b5 -> b5
    .quantity(2)
    .items(Method.HEAD,
    Method.GET))))
    .cacheBehaviors(b -> b
    .quantity(1)
    .items(b2 -> b2

```

```
.pathPattern("/index.html")

.viewerProtocolPolicy(
    ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3
    .quantity(1)
    .items(keyGroupId)
    .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4
    .cookies(cp -> cp
        .forward(ItemSelection.NONE))
    .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)
    .items(Method.HEAD,
        Method.GET)
    .cachedMethods(b6 -> b6
        .quantity(2)
        .items(Method.HEAD,
            Method.GET))))
    .enabled(true)
    .comment("Distribution built with
java")
```



```

.callerReference(Instant.now().toString()));

        final Distribution distribution = createDistResponse.distribution();
        logger.info("Distribution created. DomainName: [{}] Id: [{}]",
distribution.domainName(),
                        distribution.id());
        logger.info("Waiting for distribution to be deployed ...");
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                                .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))
                                .matched();
            responseOrException.response()
                                .orElseThrow(() -> new
RuntimeException("Distribution not created"));
            logger.info("Distribution deployed. DomainName: [{}] Id:
[{}]", distribution.domainName(),
                                distribution.id());
        }
        return distribution;
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateDistribution](#) à la section Référence des AWS SDK for Java 2.x API.

CreateFunction

L'exemple de code suivant montre comment utiliser `CreateFunction`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName> <filePath>

            Where:
                functionName - The name of the function to create.\s
                filePath - The path to a file that contains the application
            logic for the function.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
            .region(Region.AWS_GLOBAL)
            .build();
```

```
String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
System.out.println("The function ARN is " + funArn);
cloudFrontClient.close();
}

public static String createNewFunction(CloudFrontClient cloudFrontClient, String
functionName, String filePath) {
    try {
        InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
        SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

        FunctionConfig config = FunctionConfig.builder()
            .comment("Created by using the CloudFront Java API")
            .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .name(functionName)
            .functionCode(functionCode)
            .functionConfig(config)
            .build();

        CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
        return response.functionSummary().functionMetadata().functionARN();

    } catch (CloudFrontException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateFunction](#) à la section Référence des AWS SDK for Java 2.x API.

CreateKeyGroup

L'exemple de code suivant montre comment utiliser CreateKeyGroup.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Un groupe de clés nécessite au moins une clé publique utilisée pour vérifier les URL signées ou les cookies.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
        LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b -> b.keyGroupConfig(c
-> c
            .items(publicKeyId)
            .name("JavaKeyGroup" + UUID.randomUUID()))
            .keyGroup().id());
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyGroup](#) à la section Référence des AWS SDK for Java 2.x API.

CreatePublicKey

L'exemple de code suivant montre comment utiliser `CreatePublicKey`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

L'exemple de code suivant lit une clé publique et la télécharge sur Amazon CloudFront.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
        LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
        publicKeyFileName) {
        try (InputStream is =
            CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                .createPublicKey(b -> b.publicKeyConfig(c -> c
                    .name("JavaCreatedPublicKey" + UUID.randomUUID())
                    .encodedKey(publicKeyString)
                    .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreatePublicKey](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteDistribution

L'exemple de code suivant montre comment utiliser `DeleteDistribution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

L'exemple de code suivant met une distribution à Disabled, utilise un serveur qui attend que la modification soit déployée, puis supprime la distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
    private static final Logger logger =
        LoggerFactory.getLogger(DeleteDistribution.class);

    public static void deleteDistribution(final CloudFrontClient
        cloudFrontClient, final String distributionId) {
        // First, disable the distribution by updating it.
        GetDistributionResponse response =
            cloudFrontClient.getDistribution(b -> b
                .id(distributionId));
        String etag = response.eTag();
        DistributionConfig distConfig =
            response.distribution().distributionConfig();
```

```

        cloudFrontClient.updateDistribution(builder -> builder
            .id(distributionId)
            .distributionConfig(builder1 -> builder1

                .cacheBehaviors(distConfig.cacheBehaviors())

                .defaultCacheBehavior(distConfig.defaultCacheBehavior())
                    .enabled(false)
                    .origins(distConfig.origins())
                    .comment(distConfig.comment())

                .callerReference(distConfig.callerReference())

                .defaultCacheBehavior(distConfig.defaultCacheBehavior())
                    .priceClass(distConfig.priceClass())
                    .aliases(distConfig.aliases())
                    .logging(distConfig.logging())

                .defaultRootObject(distConfig.defaultRootObject())

                .customErrorResponses(distConfig.customErrorResponses())

                .httpVersion(distConfig.httpVersion())

                .isIPV6Enabled(distConfig.isIPV6Enabled())

                .restrictions(distConfig.restrictions())

                .viewerCertificate(distConfig.viewerCertificate())
                    .webACLId(distConfig.webACLId())

                .originGroups(distConfig.originGroups())
                    .ifMatch(etag));

        logger.info("Distribution [{}] is DISABLED, waiting for deployment
before deleting ...",
            distributionId);
        GetDistributionResponse distributionResponse;
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter

```

```

        .waitUntilDistributionDeployed(builder ->
builder.id(distributionId)).matched();
        distributionResponse = responseOrException.response()
            .orElseThrow(() -> new
RuntimeException("Could not disable distribution"));
    }

    DeleteDistributionResponse deleteDistributionResponse =
cloudFrontClient
        .deleteDistribution(builder -> builder
            .id(distributionId)

        .ifMatch(distributionResponse.eTag()));
        if (deleteDistributionResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Distribution [{}] DELETED", distributionId);
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [DeleteDistribution](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateDistribution

L'exemple de code suivant montre comment utiliser `UpdateDistribution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;

```



```
import software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <id>\s

                Where:
                id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                    .id(idVal)
                    .build();
```

```
    GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
    Distribution disObject = response.distribution();
    DistributionConfig config = disObject.distributionConfig();

    // Create a new DistributionConfig object and add new values to comment
and
    // aliases
    DistributionConfig config1 = DistributionConfig.builder()
        .aliases(config.aliases()) // You can pass in new values here
        .comment("New Comment")
        .cacheBehaviors(config.cacheBehaviors())
        .priceClass(config.priceClass())
        .defaultCacheBehavior(config.defaultCacheBehavior())
        .enabled(config.enabled())
        .callerReference(config.callerReference())
        .logging(config.logging())
        .originGroups(config.originGroups())
        .origins(config.origins())
        .restrictions(config.restrictions())
        .defaultRootObject(config.defaultRootObject())
        .webACLId(config.webACLId())
        .httpVersion(config.httpVersion())
        .viewerCertificate(config.viewerCertificate())
        .customErrorResponses(config.customErrorResponses())
        .build();

    UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
        .distributionConfig(config1)
        .id(disObject.id())
        .ifMatch(response.eTag())
        .build();

    cloudFrontClient.updateDistribution(updateDistributionRequest);

} catch (CloudFrontException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateDistribution](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Supprimer les ressources de signature

L'exemple de code suivant montre comment supprimer des ressources utilisées pour accéder à du contenu restreint dans un compartiment Amazon Simple Storage Service (Amazon S3).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
        LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
        cloudFrontClient,
        final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
            .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
            cloudFrontClient.deleteOriginAccessControl(builder -> builder
```

```

        .id(originAccessControlId)
        .ifMatch(getResponse.eTag()));
    if (deleteResponse.sdkHttpResponse().isSuccessful()) {
        logger.info("Successfully deleted Origin Access Control [{}]",
originAccessControlId);
    }
}

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
cloudFrontClient.deleteKeyGroup(builder -> builder
            .id(keyGroupId)
            .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
cloudFrontClient.deletePublicKey(builder -> builder
            .id(publicKeyId)
            .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}

```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [DeleteKeyGroup](#)

- [DeleteOriginAccessControl](#)
- [DeletePublicKey](#)

URL de signature et cookies

L'exemple de code suivant montre comment créer des URL signées et des cookies qui permettent d'accéder à des ressources restreintes.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez la [CannedSignerRequest](#) classe pour signer des URL ou des cookies avec une politique prédéfinie.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);
```

```

        return CannedSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expirationDate)
            .build();
    }
}

```

Utilisez la [CustomSignerRequest](#) classe pour signer des URL ou des cookies avec une politique personnalisée. Les `activeDate` et `ipRange` sont des méthodes facultatives.

```

import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expireDate)
            .activeDate(activeDate) // Optional.

```

```

        // .ipRange("192.168.0.1/24") // Optional.
        .build();
    }
}

```

L'exemple suivant illustre l'utilisation de la [CloudFrontUtilities](#) classe pour produire des cookies et des URL signés. [Consultez](#) cet exemple de code sur GitHub.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
        LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
        CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
cannedSignerRequest) {
        SignedUrl signedUrl =
cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
customSignerRequest) {
        SignedUrl signedUrl =
cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities

```

```

        .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
            .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [CloudFrontUtilities](#) à la section Référence des AWS SDK for Java 2.x API.

CloudWatch exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with CloudWatch.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.


Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour CloudWatch

Les exemples de code suivants montrent comment commencer à utiliser CloudWatch.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
                EC2).\s

                """;
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String namespace = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    listMets(cw, namespace);
    cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> System.out.println(" Retrieved metric is: "
+ metrics.metricName()));

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListMetrics](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

DeleteAlarms

L'exemple de code suivant montre comment utiliser `DeleteAlarms`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alarmName>

                Where:
                alarmName - An alarm name to delete (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String alarmName = args[0];
Region region = Region.US_EAST_2;
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .build();

deleteCWAlarm(cw, alarmName);
cw.close();
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAlarms](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteAnomalyDetector

L'exemple de code suivant montre comment utiliser `DeleteAnomalyDetector`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAnomalyDetector](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteDashboards

L'exemple de code suivant montre comment utiliser `DeleteDashboards`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");


    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDashboards](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeAlarmHistory

L'exemple de code suivant montre comment utiliser `DescribeAlarmHistory`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
            .alarmName(alarmName)
            .historyItemType(HistoryItemType.ACTION)
            .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
                System.out.println("History summary: " + item.historySummary());
                System.out.println("Time stamp: " + item.timestamp());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAlarmHistory](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeAlarms

L'exemple de code suivant montre comment utiliser `DescribeAlarms`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
            System.out.println("Alarm description: " +
alarm.alarmDescription());
        }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAlarms](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeAlarmsForMetric

L'exemple de code suivant montre comment utiliser `DescribeAlarmsForMetric`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
```

```
        System.out.println("Alarm state found for " + customMetricName +
        ".");
    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAlarmsForMetric](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeAnomalyDetectors

L'exemple de code suivant montre comment utiliser `DescribeAnomalyDetectors`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
```

```

        .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeAnomalyDetectors](#) à la section Référence des AWS SDK for Java 2.x API.

DisableAlarmActions

L'exemple de code suivant montre comment utiliser `DisableAlarmActions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <alarmName>

            Where:
                alarmName - An alarm name to disable (for example, MyAlarm).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
            .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
DisableAlarmActionsRequest.builder()
                .alarmNames(alarmName)
                .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DisableAlarmActions](#) à la section Référence des AWS SDK for Java 2.x API.

EnableAlarmActions

L'exemple de code suivant montre comment utiliser `EnableAlarmActions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alarmName>
```

```
        Where:
        alarmName - An alarm name to enable (for example, MyAlarm).
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String alarm = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    enableActions(cw, alarm);
    cw.close();
}

public static void enableActions(CloudWatchClient cw, String alarm) {
    try {
        EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
            .alarmNames(alarm)
            .build();

        cw.enableAlarmActions(request);
        System.out.printf("Successfully enabled actions on alarm %s", alarm);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [EnableAlarmActions](#) à la section Référence des AWS SDK for Java 2.x API.

GetMetricData

L'exemple de code suivant montre comment utiliser `GetMetricData`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
            .id("foo2")
            .returnData(true)
```

```
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetMetricData](#) à la section Référence des AWS SDK for Java 2.x API.

GetMetricStatistics

L'exemple de code suivant montre comment utiliser `GetMetricStatistics`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
        String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetMetricStatistics](#) à la section Référence des AWS SDK for Java 2.x API.

GetMetricWidgetImage

L'exemple de code suivant montre comment utiliser `GetMetricWidgetImage`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\n" +
            "  \"view\": \"timeSeries\",\n" +
            "  \"stacked\": false,\n" +
            "  \"period\": 10,\n" +
            "  \"width\": 1400,\n" +
            "  \"height\": 600,\n" +
            "  \"metrics\": [\n" +
            "    [\n" +
            "      \"AWS/Billing\",\n" +
            "      \"EstimatedCharges\",\n" +
            "      \"Currency\",\n" +
            "      \"USD\"\n" +
            "    ]\n" +
            "  ]\n" +
            "}";

        GetMetricWidgetImageRequest imageRequest =
            GetMetricWidgetImageRequest.builder()
                .metricWidget(myJSON)
                .build();

        GetMetricWidgetImageResponse response =
            cw.getMetricWidgetImage(imageRequest);
        SdkBytes sdkBytes = response.metricWidgetImage();
        byte[] bytes = sdkBytes.asByteArray();
        File outputFile = new File(fileName);
        try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
```

```
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetMetricWidgetImage](#) à la section Référence des AWS SDK for Java 2.x API.

ListDashboards

L'exemple de code suivant montre comment utiliser `ListDashboards`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDashboards](#) à la section Référence des AWS SDK for Java 2.x API.

ListMetrics

L'exemple de code suivant montre comment utiliser `ListMetrics`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
```

```
namespace - The namespace to filter against (for example, AWS/
EC2).\s
    """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String namespace = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    listMets(cw, namespace);
    cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    boolean done = false;
    String nextToken = null;

    try {
        while (!done) {

            ListMetricsResponse response;
            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .nextToken(nextToken)
                    .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf("Retrieved metric %s", metric.metricName());
            }
        }
    }
}
```

```
        System.out.println();
    }

    if (response.nextToken() == null) {
        done = true;
    } else {
        nextToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListMetrics](#) à la section Référence des AWS SDK for Java 2.x API.

PutAnomalyDetector

L'exemple de code suivant montre comment utiliser `PutAnomalyDetector`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
String customMetricName =
rootNode.findValue("customMetricName").asText();

SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .stat("Maximum")
    .build();

PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
    .build();

cw.putAnomalyDetector(anomalyDetectorRequest);
System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutAnomalyDetector](#) à la section Référence des AWS SDK for Java 2.x API.

PutDashboard

L'exemple de code suivant montre comment utiliser PutDashboard.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutDashboard](#) à la section Référence des AWS SDK for Java 2.x API.

PutMetricAlarm

L'exemple de code suivant montre comment utiliser `PutMetricAlarm`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;
    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }
```

- Pour plus de détails sur l'API, reportez-vous [PutMetricAlarm](#) à la section Référence des AWS SDK for Java 2.x API.

PutMetricData

L'exemple de code suivant montre comment utiliser `PutMetricData`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {  
    try {  
        // Read values from the JSON file.  
        JsonParser parser = new JsonFactory().createParser(new File(fileName));  
        com.fasterxml.jackson.databind.JsonNode rootNode = new  
ObjectMapper().readTree(parser);  
        String customMetricNamespace =  
rootNode.findValue("customMetricNamespace").asText();  
        String customMetricName =  
rootNode.findValue("customMetricName").asText();  
  
        // Set an Instant object.  
        String time =  
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);  
        Instant instant = Instant.parse(time);  
  
        MetricDatum datum = MetricDatum.builder()  
            .metricName(customMetricName)  
            .unit(StandardUnit.NONE)  
            .value(1001.00)  
            .timestamp(instant)  
            .build();
```

```
        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
            .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
            customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutMetricData](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Démarrage avec les métriques, tableaux de bord et alertes

L'exemple de code suivant illustre comment :

- CloudWatch Répertoriez les espaces de noms et les métriques.
- obtenir les statistiques d'une métrique et de la facturation estimée ;
- créer et mettre à jour un tableau de bord ;
- créer et ajouter des données à une métrique ;

- créer et déclencher une alerte, puis consulter l'historique des alertes ;
- créer un détecteur d'anomalies ;
- obtenez une image de métrique, puis nettoyer les ressources.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
```

```
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor\_estimated\_charges\_with\_cloudwatch.html#turning\_on\_billing\_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

                Usage:
```

```

    <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

```

Where:

myDate - The start date to use to get metric statistics. (For example, 2023-01-11T18:35:24.00Z.)\s

costDateWeek - The start date to use to get AWS/Billinget statistics. (For example, 2023-01-11T18:35:24.00Z.)\s

dashboardName - The name of the dashboard to create.\s

dashboardJson - The location of a JSON file to use to create a dashboard. (See Readme file.)\s

dashboardAdd - The location of a JSON file to use to update a dashboard. (See Readme file.)\s

settings - The location of a JSON file from which various values are read. (See Readme file.)\s

metricImage - The location of a BMP file that is used to create a graph.\s

```

    """;

```

```

if (args.length != 7) {
    System.out.println(usage);
    System.exit(1);
}

```

```

Region region = Region.US_EAST_1;
String myDate = args[0];
String costDateWeek = args[1];
String dashboardName = args[2];
String dashboardJson = args[3];
String dashboardAdd = args[4];
String settings = args[5];
String metricImage = args[6];

```

```

Double dataPoint = Double.parseDouble("10.0");
Scanner sc = new Scanner(System.in);
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

```

```

System.out.println(DASHES);
System.out.println("Welcome to the Amazon CloudWatch example scenario.");
System.out.println(DASHES);

```

```
System.out.println(DASHES);
System.out.println(
    "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
ArrayList<String> list = listNameSpaces(cw);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + list.get(z));
}

String selectedNamespace = "";
String selectedMetrics = "";
int num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedNamespace = list.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedNamespace);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List available metrics within the selected namespace
and select one from the list.");
ArrayList<String> metList = listMets(cw, selectedNamespace);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + metList.get(z));
}
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedMetrics = metList.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedMetrics);
Dimension myDimension = getSpecificMet(cw, selectedNamespace);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get statistics for the selected metric over the last
day.");
```



```
String metricOption = "";
ArrayList<String> statTypes = new ArrayList<>();
statTypes.add("SampleCount");
statTypes.add("Average");
statTypes.add("Sum");
statTypes.add("Minimum");
statTypes.add("Maximum");

for (int t = 0; t < 5; t++) {
    System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
}
System.out.println("Select a metric statistic by entering a number from the
preceding list:");
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    metricOption = statTypes.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + metricOption);
getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get CloudWatch estimated billing for the last
week.");
getMetricStatistics(cw, costDateWeek);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new CloudWatch dashboard with metrics.");
createDashboardWithMetrics(cw, dashboardName, dashboardJson);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List dashboards using a paginator.");
listDashboards(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a new custom metric by adding data to it.");
createNewCustomMetric(cw, dataPoint);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Add an additional metric to the dashboard.");
addMetricToDashboard(cw, dashboardAdd, dashboardName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Create an alarm for the custom metric.");
String alarmName = createAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Describe ten current alarms.");
describeAlarms(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Get current data for new custom metric.");
getCustomMetricData(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Push data into the custom metric to trigger the
alarm.");
addMetricDataForAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
checkForMetricAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Get alarm history for the new alarm.");
getAlarmHistory(cw, settings, myDate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Add an anomaly detector for the custom metric.");
addAnomalyDetector(cw, settings);
System.out.println(DASHES);
```

```

        System.out.println(DASHES);
        System.out.println("16. Describe current anomaly detectors.");
        describeAnomalyDetectors(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is complete.");
        System.out.println(DASHES);
        cw.close();
    }

    public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)

```

```
        .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.println("Successfully deleted alarm " + alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
```

```

System.out.println("Getting Image data for custom metric.");
try {
    String myJSON = "{\n" +
        "  \"title\": \"Example Metric Graph\",\n" +
        "  \"view\": \"timeSeries\",\n" +
        "  \"stacked\": false,\n" +
        "  \"period\": 10,\n" +
        "  \"width\": 1400,\n" +
        "  \"height\": 600,\n" +
        "  \"metrics\": [\n" +
        "    [\n" +
        "      \"AWS/Billing\",\n" +
        "      \"EstimatedCharges\",\n" +
        "      \"Currency\",\n" +
        "      \"USD\"\n" +
        "    ]\n" +
        "  ]\n" +
        "}";

    GetMetricWidgetImageRequest imageRequest =
    GetMetricWidgetImageRequest.builder()
        .metricWidget(myJSON)
        .build();

    GetMetricWidgetImageResponse response =
    cw.getMetricWidgetImage(imageRequest);
    SdkBytes sdkBytes = response.metricWidgetImage();
    byte[] bytes = sdkBytes.asByteArray();
    File outputFile = new File(fileName);
    try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));

```

```
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
```

```
        .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
        .startDate(start)
        .endDate(endDate)
        .alarmName(alarmName)
        .historyItemType(HistoryItemType.ACTION)
        .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
```

```
        System.out.println("History summary: " + item.historySummary());
        System.out.println("Time stamp: " + item.timestamp());
    }
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
            System.out.println("Alarm state found for " + customMetricName +
".");
    }
}
```



```
    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
```

```
        .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
```

```
        .id("foo2")
        .returnData(true)
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
        }
    }
}
```

```
        System.out.println("Alarm description: " +
alarm.alarmDescription());
    }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

        .comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();
```

```
        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully updated.");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint)
{
    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
```

```
        .value(dataPoint)
        .timestamp(instant)
        .dimensions(dimension)
        .build();

    PutMetricDataRequest request = PutMetricDataRequest.builder()
        .namespace("SITE/TRAFFIC")
        .metricData(datum)
        .build();

    cw.putMetricData(request);
    System.out.println("Added metric values for for metric PAGES_VISITED");

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();
```

```
        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String readFileAsString(String file) throws IOException {
    return new String(Files.readAllBytes(Paths.get(file)));
}

public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek)
{
    try {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
            .statistics(Statistic.MAXIMUM)
            .startTime(start)
            .endTime(endDate)
            .period(86400)
    }
```

```

        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
        String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400)
            .statistics(Statistic.fromValue(metricOption))
            .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {

```



```
        System.out
            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
    }
    } else {
        System.out.println("The returned data list is empty");
    }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static Dimension getSpecificMet(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsResponse response = cw.listMetrics(request);
        List<Metric> myList = response.metrics();
        Metric metric = myList.get(0);
        return metric.dimensions().get(0);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listMets(CloudWatchClient cw, String namespace)
{
    try {
        ArrayList<String> metList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> metList.add(metrics.metricName()));
    }
}
```

```
        return metList;

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
    try {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> {
                String data = metrics.namespace();
                if (!nameSpaceList.contains(data)) {
                    nameSpaceList.add(data);
                }
            });

        return nameSpaceList;
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [DeleteAlarms](#)
 - [DeleteAnomalyDetector](#)
 - [DeleteDashboards](#)
 - [DescribeAlarmHistory](#)

- [DescribeAlarms](#)
- [DescribeAlarmsForMetric](#)
- [DescribeAnomalyDetectors](#)
- [GetMetricData](#)
- [GetMetricStatistics](#)
- [GetMetricWidgetImage](#)
- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

CloudWatch Exemples d'événements utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide AWS SDK for Java 2.x des CloudWatch événements with.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

PutEvents

L'exemple de code suivant montre comment utiliserPutEvents.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <resourceArn>

                Where:
                resourceArn - An Amazon Resource Name (ARN) related to the
events.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();
```

```
        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                .detail(EVENT_DETAILS)
                .detailType("sampleSubmitted")
                .resources(resourceArn)
                .source("aws-sdk-java-cloudwatch-example")
                .build();

            PutEventsRequest request = PutEventsRequest.builder()
                .entries(requestEntry)
                .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutEvents](#) à la section Référence des AWS SDK for Java 2.x API.

PutRule

L'exemple de code suivant montre comment utiliser `PutRule`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <ruleName> roleArn\s

            Where:
                ruleName - A rule name (for example, myrule).
                roleArn - A role ARN value (for example,
arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
```

```
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
            .build();

        putCWRule(cwe, ruleName, roleArn);
        cwe.close();
    }

    public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {
        try {
            PutRuleRequest request = PutRuleRequest.builder()
                .name(ruleName)
                .roleArn(roleArn)
                .scheduleExpression("rate(5 minutes)")
                .state(RuleState.ENABLED)
                .build();

            PutRuleResponse response = cwe.putRule(request);
            System.out.printf(
                "Successfully created CloudWatch events rule %s with arn %s",
                roleArn, response.ruleArn());

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutRule](#) à la section Référence des AWS SDK for Java 2.x API.

PutTargets

L'exemple de code suivant montre comment utiliser `PutTargets`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <ruleName> <functionArn> <targetId>\s

            Where:
                ruleName - A rule name (for example, myrule).
                functionArn - An AWS Lambda function ARN (for example,
                arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                targetId - A target id value.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
```



```
String targetId = args[2];
CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
    .build();

putCWTargets(cwe, ruleName, functionArn, targetId);
cwe.close();
}

public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
String functionArn, String targetId) {
    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutTargets](#) à la section Référence des AWS SDK for Java 2.x API.

CloudWatch Exemples de journaux utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK for Java 2.x with CloudWatch Logs.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

DeleteSubscriptionFilter

L'exemple de code suivant montre comment utiliser DeleteSubscriptionFilter.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <filter> <logGroup>

            Where:
                filter - The name of the subscription filter (for example,
MyFilter).
                logGroup - The name of the log group. (for example, testgroup).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
            .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
                .filterName(filter)
                .logGroupName(logGroup)
                .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
filter %s", filter);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSubscriptionFilter](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeSubscriptionFilters

L'exemple de code suivant montre comment utiliser `DescribeSubscriptionFilters`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;  
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;  
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;  
import  
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;  
import  
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;  
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DescribeSubscriptionFilters {  
    public static void main(String[] args) {  
  
        final String usage = ""
```

```
Usage:
  <logGroup>

Where:
  logGroup - A log group name (for example, myloggroup).
  """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String logGroup = args[0];
CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

describeFilters(logs, logGroup);
logs.close();
}

public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
    try {
        boolean done = false;
        String newToken = null;

        while (!done) {
            DescribeSubscriptionFiltersResponse response;
            if (newToken == null) {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .logGroupName(logGroup)
                    .limit(1).build();

                response = logs.describeSubscriptionFilters(request);
            } else {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .nextToken(newToken)
                    .logGroupName(logGroup)
                    .limit(1).build();
                response = logs.describeSubscriptionFilters(request);
            }
        }
    }
}
```

```
        for (SubscriptionFilter filter : response.subscriptionFilters()) {
            System.out.printf("Retrieved filter with name %s, " + "pattern
%s " + "and destination arn %s",
                filter.filterName(),
                filter.filterPattern(),
                filter.destinationArn());
        }

        if (response.nextToken() == null) {
            done = true;
        } else {
            newToken = response.nextToken();
        }
    }

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSubscriptionFilters](#) à la section Référence des AWS SDK for Java 2.x API.

PutSubscriptionFilter

L'exemple de code suivant montre comment utiliser `PutSubscriptionFilter`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
```

```
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 * Also, before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <filter> <pattern> <logGroup> <functionArn>\s

            Where:
                filter - A filter name (for example, myfilter).
                pattern - A filter pattern (for example, ERROR).
                logGroup - A log group name (testgroup).
                functionArn - An AWS Lambda function ARN (for example,
                arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
            """;
    }
}
```

```
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String filter = args[0];
    String pattern = args[1];
    String logGroup = args[2];
    String functionArn = args[3];
    Region region = Region.US_WEST_2;
    CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
        .region(region)
        .build();

    putSubFilters(cwl, filter, pattern, logGroup, functionArn);
    cwl.close();
}

public static void putSubFilters(CloudWatchLogsClient cwl,
    String filter,
    String pattern,
    String logGroup,
    String functionArn) {

    try {
        PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
            .filterName(filter)
            .filterPattern(pattern)
            .logGroupName(logGroup)
            .destinationArn(functionArn)
            .build();

        cwl.putSubscriptionFilter(request);
        System.out.printf(
            "Successfully created CloudWatch logs subscription filter %s",
            filter);

    } catch (CloudWatchLogsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```


- Pour plus de détails sur l'API, reportez-vous [PutSubscriptionFilter](#) à la section Référence des AWS SDK for Java 2.x API.

StartLiveTail

L'exemple de code suivant montre comment utiliser `StartLiveTail`.

SDK pour Java 2.x

Joignez les fichiers requis.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Gérez les événements de la session Live Tail.

```
private static StartLiveTailResponseHandler
getStartLiveTailResponseStreamHandler(
    AtomicReference<Subscription> subscriptionAtomicReference) {
    return StartLiveTailResponseHandler.builder()
        .onResponse(r -> System.out.println("Received initial response"))
        .onError(throwable -> {
```

```

        CloudWatchLogsException e = (CloudWatchLogsException)
throwable.getCause();
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    })
    .subscriber(() -> new FlowableSubscriber<>() {
        @Override
        public void onSubscribe(@NonNull Subscription s) {
            subscriptionAtomicReference.set(s);
            s.request(Long.MAX_VALUE);
        }

        @Override
        public void onNext(StartLiveTailResponseStream event) {
            if (event instanceof LiveTailSessionStart) {
                LiveTailSessionStart sessionStart = (LiveTailSessionStart)
event;

                System.out.println(sessionStart);
            } else if (event instanceof LiveTailSessionUpdate) {
                LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                logEvents.forEach(e -> {
                    long timestamp = e.timestamp();
                    Date date = new Date(timestamp);
                    System.out.println "[" + date + "]" + e.message());
                });
            } else {
                throw CloudWatchLogsException.builder().message("Unknown
event type").build();
            }
        }

        @Override
        public void onError(Throwable throwable) {
            System.out.println(throwable.getMessage());
            System.exit(1);
        }

        @Override
        public void onComplete() {
            System.out.println("Completed Streaming Session");
        }
    }

```

```
    })
    .build();
}
```

Démarrez la session Live Tail.

```
CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
    CloudWatchLogsAsyncClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

StartLiveTailRequest request =
    StartLiveTailRequest.builder()
        .logGroupIdentifiers(logGroupIdentifiers)
        .logStreamNames(logStreamNames)
        .logEventFilterPattern(logEventFilterPattern)
        .build();

/* Create a reference to store the subscription */
final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

Arrêtez la session Live Tail au bout d'un certain temps.

```
/* Set a timeout for the session and cancel the subscription. This will:
 * 1). Close the stream
 * 2). Stop the Live Tail session
 */
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
if (subscriptionAtomicReference.get() != null) {
    subscriptionAtomicReference.get().cancel();
    System.out.println("Subscription to stream closed");
}
```

- Pour plus de détails sur l'API, reportez-vous [StartLiveTail](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'Amazon Cognito Identity utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Cognito Identity.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateIdentityPool

L'exemple de code suivant montre comment utiliser `CreateIdentityPool`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <identityPoolName>\s

            Where:
                identityPoolName - The name to give your identity pool.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
identityPoolName) {
        try {
```

```
        CreateIdentityPoolRequest poolRequest =
CreateIdentityPoolRequest.builder()
    .allowUnauthenticatedIdentities(false)
    .identityPoolName(identityPoolName)
    .build();

        CreateIdentityPoolResponse response =
cognitoClient.createIdentityPool(poolRequest);
        return response.identityPoolId();

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateIdentityPool](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteIdentityPool

L'exemple de code suivant montre comment utiliser `DeleteIdentityPool`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <identityPoolId>\s

            Where:
                identityPoolId - The Id value of your identity pool.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolId = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        deleteIdPool(cognitoIdClient, identityPoolId);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
identityPoolId) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
DeleteIdentityPoolRequest.builder()
                .identityPoolId(identityPoolId)
                .build();
```

```
        cognitoIdClient.deleteIdentityPool(identityPoolRequest);
        System.out.println("Done");

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteIdentityPool](#) à la section Référence des AWS SDK for Java 2.x API.

GetCredentialsForIdentity

L'exemple de code suivant montre comment utiliser `GetCredentialsForIdentity`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```



```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
GetCredentialsForIdentityRequest
                .builder()
                .identityId(identityId)
                .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                "Identity ID " + response.identityId() + ", Access key ID " +
response.credentials().accessKeyId());
        }
    }
}
```

```
        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetCredentialsForIdentity](#) à la section Référence des AWS SDK for Java 2.x API.

ListIdentityPools

L'exemple de code suivant montre comment utiliser `ListIdentityPools`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
```

```
public static void main(String[] args) {
    CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
        .region(Region.US_EAST_1)
        .build();

    listIdPools(cognitoClient);
    cognitoClient.close();
}

public static void listIdPools(CognitoIdentityClient cognitoClient) {
    try {
        ListIdentityPoolsRequest poolsRequest =
ListIdentityPoolsRequest.builder()
        .maxResults(15)
        .build();

        ListIdentityPoolsResponse response =
cognitoClient.listIdentityPools(poolsRequest);
        response.identityPools().forEach(pool -> {
            System.out.println("Pool ID: " + pool.identityPoolId());
            System.out.println("Pool name: " + pool.identityPoolName());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListIdentityPools](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples de fournisseurs d'identité Amazon Cognito utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du fournisseur AWS SDK for Java 2.x d'identité Amazon Cognito.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Amazon Cognito

Les exemples de code suivants montrent comment bien démarrer avec Amazon Cognito.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                .maxResults(10)
                .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListUserPools](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

AdminGetUser

L'exemple de code suivant montre comment utiliser `AdminGetUser`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
    String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [AdminGetUser](#) à la section Référence des AWS SDK for Java 2.x API.

AdminInitiateAuth

L'exemple de code suivant montre comment utiliser `AdminInitiateAuth`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [AdminInitiateAuth](#) à la section Référence des AWS SDK for Java 2.x API.

AdminRespondToAuthChallenge

L'exemple de code suivant montre comment utiliser `AdminRespondToAuthChallenge`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
    String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}
```

- Pour plus de détails sur l'API, reportez-vous [AdminRespondToAuthChallenge](#) à la section Référence des AWS SDK for Java 2.x API.

AssociateSoftwareToken

L'exemple de code suivant montre comment utiliser `AssociateSoftwareToken`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

- Pour plus de détails sur l'API, reportez-vous [AssociateSoftwareToken](#) à la section Référence des AWS SDK for Java 2.x API.

ConfirmSignUp

L'exemple de code suivant montre comment utiliser `ConfirmSignUp`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ConfirmSignUp](#) à la section Référence des AWS SDK for Java 2.x API.

CreateUserPool

L'exemple de code suivant montre comment utiliser `CreateUserPool`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolName>\s

            Where:
                userPoolName - The name to give your user pool when it's
created.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String userPoolName = args[0];
    CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String id = createPool(cognitoClient, userPoolName);
    System.out.println("User pool ID: " + id);
    cognitoClient.close();
}

public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
    try {
        CreateUserPoolRequest request = CreateUserPoolRequest.builder()
            .poolName(userPoolName)
            .build();

        CreateUserPoolResponse response = cognitoClient.createUserPool(request);
        return response.userPool().id();

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateUserPool](#) à la section Référence des AWS SDK for Java 2.x API.

CreateUserPoolClient

L'exemple de code suivant montre comment utiliser `CreateUserPoolClient`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clientName> <userPoolId>\s

                Where:
                clientName - The name for the user pool client to create.
```

```
        userPoolId - The ID for the user pool.
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String clientName = args[0];
    String userPoolId = args[1];
    CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPoolClient(cognitoClient, clientName, userPoolId);
    cognitoClient.close();
}

public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
String clientName,
    String userPoolId) {
    try {
        CreateUserPoolClientRequest request =
CreateUserPoolClientRequest.builder()
            .clientName(clientName)
            .userPoolId(userPoolId)
            .build();

        CreateUserPoolClientResponse response =
cognitoClient.createUserPoolClient(request);
        System.out.println("User pool " + response.userPoolClient().clientName()
+ " created. ID: "
            + response.userPoolClient().clientId());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateUserPoolClient](#) à la section Référence des AWS SDK for Java 2.x API.

ListUserPools

L'exemple de code suivant montre comment utiliser `ListUserPools`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
        CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }
}
```

```
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                .maxResults(10)
                .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
                    userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListUserPools](#) à la section Référence des AWS SDK for Java 2.x API.

ListUsers

L'exemple de code suivant montre comment utiliser `ListUsers`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
```



```
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolId>\s

            Where:
                userPoolId - The ID given to your user pool when it's created.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }
}
```

```
public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
    try {
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Shows how to list users by using a filter.
public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {

    try {
        String filter = "email = \"tblue@noserver.com\"";
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .filter(filter)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User with filter applied " + user.username() + "
Status " + user.userStatus()
                + " Created " + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListUsers](#) à la section Référence des AWS SDK for Java 2.x API.

ResendConfirmationCode

L'exemple de code suivant montre comment utiliser `ResendConfirmationCode`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ResendConfirmationCode](#) à la section Référence des AWS SDK for Java 2.x API.

SignUp

L'exemple de code suivant montre comment utiliser `SignUp`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SignUp](#) à la section Référence des AWS SDK for Java 2.x API.

VerifySoftwareToken

L'exemple de code suivant montre comment utiliser `VerifySoftwareToken`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [VerifySoftwareToken](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Inscription d'un utilisateur auprès d'un groupe d'utilisateurs nécessitant l'authentification MFA

L'exemple de code suivant illustre comment :

- Inscrivez et confirmez un utilisateur avec un nom d'utilisateur, un mot de passe et une adresse e-mail.
- Configurez l'authentification multifactorielle en associant une application MFA à l'utilisateur.
- Connectez-vous à l'aide d'un mot de passe et d'un code MFA.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
```

```
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito\_scenario\_user\_pool\_with\_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
```

```

* "ChallengeName": "MFA_SETUP").
* 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
* key. This can be used with Google Authenticator.
* 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
* MFA.
* 8. Invokes the AdminInitiateAuth to sign in again. This results in being
* prompted to submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
* 9. Invokes the AdminRespondToAuthChallenge to get back a token.
*/

```

```

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
    InvalidKeyException {
        final String usage = ""

            Usage:
                <clientId> <poolId>

            Where:
                clientId - The app client Id value that you can get from the AWS
CDK script.
                poolId - The pool Id that you can get from the AWS CDK script.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
    }
}

```



```
Scanner in = new Scanner(System.in);
String userName = in.nextLine();

System.out.println("*** Enter your password");
String password = in.nextLine();

System.out.println("*** Enter your email");
String email = in.nextLine();

System.out.println("1. Signing up " + userName);
signUp(identityProviderClient, clientId, userName, password, email);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Getting " + userName + " in the user pool");
getAdminUser(identityProviderClient, userName, poolId);

System.out
    .println("*** Confirmation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
System.out.println(DASHES);

System.out.println(DASHES);
String ans = in.nextLine();

if (ans.compareTo("Yes") == 0) {
    resendConfirmationCode(identityProviderClient, clientId, userName);
    System.out.println("3. Sending a new confirmation code");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enter confirmation code that was emailed");
String code = in.nextLine();
confirmSignUp(identityProviderClient, clientId, code, userName);
System.out.println("Rechecking the status of " + userName + " in the user
pool");
getAdminUser(identityProviderClient, userName, poolId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Invokes the initiateAuth to sign in");
AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
```

```
        poolId);
    String mySession = authResponse.session();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
    String newSession = getSecretForAppMFA(identityProviderClient, mySession);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
    String myCode = in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Verify the TOTP and register for MFA");
    verifyTOTP(identityProviderClient, newSession, myCode);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
    String mfaCode = in.nextLine();
    AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
        poolId);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("9. Invokes the AdminRespondToAuthChallenge");
    String session2 = authResponse1.session();
    adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("All Amazon Cognito operations were successfully
performed");
    System.out.println(DASHES);
}

// Respond to an authentication challenge.
```

```
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
    String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}

// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                                .session(session)
                                .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
```

```
AttributeType userAttrs = AttributeType.builder()
    .name("email")
    .value(email)
    .build();

List<AttributeType> userAttrsList = new ArrayList<>();
userAttrsList.add(userAttrs);
try {
    SignUpRequest signUpRequest = SignUpRequest.builder()
        .userAttributes(userAttrsList)
        .username(userName)
        .clientId(clientId)
        .password(password)
        .build();

    identityProviderClient.signUp(signUpRequest);
    System.out.println("User has been signed up ");

} catch (CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
    String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)
 - [VerifySoftwareToken](#)

Exemples d'Amazon Comprehend utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Comprehend.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateDocumentClassifier

L'exemple de code suivant montre comment utiliser `CreateDocumentClassifier`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
    software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-
amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
```



```
        dataAccessRoleArn - The ARN value of the role used for this
operation.

        s3Uri - The Amazon S3 bucket that contains the CSV file.
        documentClassifierName - The name of the document classifier.
        """";

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String dataAccessRoleArn = args[0];
    String s3Uri = args[1];
    String documentClassifierName = args[2];

    Region region = Region.US_EAST_1;
    ComprehendClient comClient = ComprehendClient.builder()
        .region(region)
        .build();

    createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
    comClient.close();
}

public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri,
    String documentClassifierName) {
    try {
        DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
            .s3Uri(s3Uri)
            .build();

        CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
            .documentClassifierName(documentClassifierName)
            .dataAccessRoleArn(dataAccessRoleArn)
            .languageCode("en")
            .inputDataConfig(config)
            .build();

        CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient
```

```
        .createDocumentClassifier(createDocumentClassifierRequest);
        String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
        System.out.println("Document Classifier ARN: " + documentClassifierArn);

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDocumentClassifier](#) à la section Référence des AWS SDK for Java 2.x API.

DetectDominantLanguage

L'exemple de code suivant montre comment utiliser `DetectDominantLanguage`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
    software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
    software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient, String
text) {
        try {
            DetectDominantLanguageRequest request =
DetectDominantLanguageRequest.builder()
                .text(text)
                .build();

            DetectDominantLanguageResponse resp =
comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectDominantLanguage](#) à la section Référence des AWS SDK for Java 2.x API.

DetectEntities

L'exemple de code suivant montre comment utiliser `DetectEntities`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();
```

```
        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    public static void detectAllEntities(ComprehendClient comClient, String text) {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
DetectEntitiesRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectEntitiesResponse detectEntitiesResult =
comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectEntities](#) à la section Référence des AWS SDK for Java 2.x API.

DetectKeyPhrases

L'exemple de code suivant montre comment utiliser DetectKeyPhrases.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String text)
    {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
            DetectKeyPhrasesRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
            comClient.detectKeyPhrases(detectKeyPhrasesRequest);
        }
    }
}
```

```
        List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
        for (KeyPhrase keyPhrase : phraseList) {
            System.out.println("Key phrase text is " + keyPhrase.text());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectKeyPhrases](#) à la section Référence des AWS SDK for Java 2.x API.

DetectSentiment

L'exemple de code suivant montre comment utiliser `DetectSentiment`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
*/
public class DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }

    public static void detectSentiments(ComprehendClient comClient, String text) {
        try {
            DetectSentimentRequest detectSentimentRequest =
            DetectSentimentRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectSentimentResponse detectSentimentResult =
            comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
            detectSentimentResult.sentimentScore().neutral());

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectSentiment](#) à la section Référence des AWS SDK for Java 2.x API.

DetectSyntax

L'exemple de code suivant montre comment utiliser DetectSyntax.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }
}
```

```
public static void detectAllSyntax(ComprehendClient comClient, String text) {
    try {
        DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectSyntaxResponse detectSyntaxResult =
comClient.detectSyntax(detectSyntaxRequest);
        List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
        for (SyntaxToken token : syntaxTokens) {
            System.out.println("Language is " + token.text());
            System.out.println("Part of speech is " +
token.partOfSpeech().tagAsString());
        }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectSyntax](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples DynamoDB utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide de DynamoDB.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Hello DynamoDB

Les exemples de code suivants montrent comment démarrer avec DynamoDB.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }
}
```

```
public static void listAllTables(DynamoDbClient ddb) {
    boolean moreTables = true;
    String lastName = null;

    while (moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
                ListTablesRequest request = ListTablesRequest.builder().build();
                response = ddb.listTables(request);
            } else {
                ListTablesRequest request = ListTablesRequest.builder()
                    .exclusiveStartTableName(lastName).build();
                response = ddb.listTables(request);
            }

            List<String> tableNames = response.tableNames();
            if (tableNames.size() > 0) {
                for (String curName : tableNames) {
                    System.out.format("* %s\n", curName);
                }
            } else {
                System.out.println("No tables found!");
                System.exit(0);
            }

            lastName = response.lastEvaluatedTableName();
            if (lastName == null) {
                moreTables = false;
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
    System.out.println("\nDone!");
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTables](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)
- [Exemples sans serveur](#)

Actions

BatchGetItem

L'exemple de code suivant montre comment utiliser `BatchGetItem`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

montre comment obtenir des articles par lots à l'aide du client de service.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchReadItems {
    public static void main(String[] args){
```

```

    final String usage = ""

        Usage:
            <tableName>

        Where:
            tableName - The Amazon DynamoDB table (for example, Music).\s
        """;

    String tableName = "Music";
    Region region = Region.US_EAST_1;
    DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
        .region(region)
        .build();

    getBatchItems(dynamoDbClient, tableName);
}

public static void getBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
    // Define the primary key values for the items you want to retrieve.
    Map<String, AttributeValue> key1 = new HashMap<>();
    key1.put("Artist", AttributeValue.builder().s("Artist1").build());

    Map<String, AttributeValue> key2 = new HashMap<>();
    key2.put("Artist", AttributeValue.builder().s("Artist2").build());

    // Construct the batchGetItem request.
    Map<String, KeysAndAttributes> requestItems = new HashMap<>();
    requestItems.put(tableName, KeysAndAttributes.builder()
        .keys(List.of(key1, key2))
        .projectionExpression("Artist, SongTitle")
        .build());

    BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
        .requestItems(requestItems)
        .build();

    // Make the batchGetItem request.
    BatchGetItemResponse batchGetItemResponse =
dynamoDbClient.batchGetItem(batchGetItemRequest);

    // Extract and print the retrieved items.

```

```

        Map<String, List<Map<String, AttributeValue>>> responses =
batchGetItemResponse.responses();
        if (responses.containsKey(tableName)) {
            List<Map<String, AttributeValue>> musicItems = responses.get(tableName);
            for (Map<String, AttributeValue> item : musicItems) {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            }
        } else {
            System.out.println("No items retrieved.");
        }
    }
}

```

montre comment obtenir des articles par lots à l'aide du client de service et d'un paginateur.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
                .region(region)

```

```

        .build();

    getBatchItemsPaginator(dynamoDbClient, tableName) ;
}

public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
tableName) {
    // Define the primary key values for the items you want to retrieve.
    Map<String, AttributeValue> key1 = new HashMap<>();
    key1.put("Artist", AttributeValue.builder().s("Artist1").build());

    Map<String, AttributeValue> key2 = new HashMap<>();
    key2.put("Artist", AttributeValue.builder().s("Artist2").build());

    // Construct the batchGetItem request.
    Map<String, KeysAndAttributes> requestItems = new HashMap<>();
    requestItems.put(tableName, KeysAndAttributes.builder()
        .keys(List.of(key1, key2))
        .projectionExpression("Artist, SongTitle")
        .build());

    BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
        .requestItems(requestItems)
        .build();

    // Use batchGetItemPaginator for paginated requests.
    dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
        .flatMap(response -> response.responses().getOrDefault(tableName,
Collections.emptyList()).stream())
        .forEach(item -> {
            System.out.println("Artist: " + item.get("Artist").s() +
                ", SongTitle: " + item.get("SongTitle").s());
        });
}
}

```

- Pour plus de détails sur l'API, reportez-vous [BatchGetItem](#) à la section Référence des AWS SDK for Java 2.x API.

BatchWriteItem

L'exemple de code suivant montre comment utiliser `BatchWriteItem`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Insère de nombreux éléments dans un tableau à l'aide du client de service.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
```

```
    Region region = Region.US_EAST_1;
    DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
        .region(region)
        .build();

    addBatchItems(dynamoDbClient, tableName);
}

public static void addBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
    // Specify the updates you want to perform.
    List<WriteRequest> writeRequests = new ArrayList<>();

    // Set item 1.
    Map<String, AttributeValue> item1Attributes = new HashMap<>();
    item1Attributes.put("Artist",
AttributeValue.builder().s("Artist1").build());
    item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
    item1Attributes.put("Comments", AttributeValue.builder().s("Great
song!").build());
    item1Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle1").build());

    writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attributes

    // Set item 2.
    Map<String, AttributeValue> item2Attributes = new HashMap<>();
    item2Attributes.put("Artist",
AttributeValue.builder().s("Artist2").build());
    item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
    item2Attributes.put("Comments", AttributeValue.builder().s("Nice
melody.").build());
    item2Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle2").build());

    writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attributes

    try {
        // Create the BatchWriteItemRequest.
        BatchWriteItemRequest batchWriteItemRequest =
BatchWriteItemRequest.builder()
            .requestItems(Map.of(tableName, writeRequests))
            .build();
```

```
        // Execute the BatchWriteItem operation.
        BatchWriteItemResponse batchWriteItemResponse =
dynamoDbClient.batchWriteItem(batchWriteItemRequest);

        // Process the response.
        System.out.println("Batch write successful: " + batchWriteItemResponse);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Insère de nombreux éléments dans une table à l'aide du client amélioré.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 * with these columns:
 * - id - the id of the record that is the key
 * - custName - the customer name
 * - email - the email value
 * - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 * credentials.
```

```
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class EnhancedBatchWriteItems {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();
        putBatchRecords(enhancedClient);
        ddb.close();
    }

    public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
        try {
            DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                TableSchema.fromBean(Customer.class));
            DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                TableSchema.fromBean(Music.class));
            LocalDate localDate = LocalDate.parse("2020-04-07");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            Customer record2 = new Customer();
            record2.setCustName("Fred Pink");
            record2.setId("id110");
            record2.setEmail("fredp@noserver.com");
            record2.setRegistrationDate(instant);

            Customer record3 = new Customer();
            record3.setCustName("Susan Pink");
            record3.setId("id120");
            record3.setEmail("spink@noserver.com");
            record3.setRegistrationDate(instant);

            Customer record4 = new Customer();
```

```

        record4.setCustName("Jerry orange");
        record4.setId("id101");
        record4.setEmail("jorange@noserver.com");
        record4.setRegistrationDate(instant);

        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
= BatchWriteItemEnhancedRequest
                                .builder()
                                .writeBatches(

WriteBatch.builder(Customer.class) // add items to the Customer

        // table

        .mappedTableResource(customerMappedTable)

        .addPutItem(builder -> builder.item(record2))

        .addPutItem(builder -> builder.item(record3))

        .addPutItem(builder -> builder.item(record4))

                                                                .build(),

WriteBatch.builder(Music.class) // delete an item from the Music

        // table

        .mappedTableResource(musicMappedTable)

        .addDeleteItem(builder -> builder.key(

            Key.builder().partitionValue(

                "Famous Band")

                .build()))

                                                                .build())

                                                                .build();

        // Add three items to the Customer table and delete one item
from the Music

        // table.

enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);

```

```
        System.out.println("done");
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [BatchWriteItem](#) à la section Référence des AWS SDK for Java 2.x API.

CreateTable

L'exemple de code suivant montre comment utiliser `CreateTable`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
Music3).
                key - The key for the Amazon DynamoDB table (for example,
Artist).

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
```

```
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
        .tableName(tableName)
        .build();

String newTable;
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTable](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteItem

L'exemple de code suivant montre comment utiliser `DeleteItem`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key> <keyval>

                Where:
                tableName - The Amazon DynamoDB table to delete the item from
                (for example, Music3).
                key - The key used in the Amazon DynamoDB table (for example,
                Artist).\s
                keyval - The key value that represents the item to delete (for
                example, Famous Band).
                """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
            .s(keyVal)
            .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
            .tableName(tableName)
            .key(keyToGet)
            .build();

        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteItem](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteTable

L'exemple de code suivant montre comment utiliser `DeleteTable`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to delete (for example,
Music3).

            **Warning** This program will delete the table that you specify!
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String tableName = args[0];
    System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    deleteDynamoDBTable(ddb, tableName);
    ddb.close();
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTable](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeTable

L'exemple de code suivant montre comment utiliser `DescribeTable`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
    software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table to get information about
                (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

String tableName = args[0];
System.out.format("Getting description for %s\n\n", tableName);
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

describeDynamoDBTable(ddb, tableName);
ddb.close();
}

public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo = ddb.describeTable(request).table();
        if (tableInfo != null) {
            System.out.format("Table name   : %s\n", tableInfo.tableName());
            System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
            System.out.format("Status      : %s\n", tableInfo.tableStatus());
            System.out.format("Item count  : %d\n", tableInfo.itemCount());
            System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

            ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
            System.out.println("Throughput");
            System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits());
            System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits());

            List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
            System.out.println("Attributes");
            for (AttributeDefinition a : attributes) {
                System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
            }
        }
    } catch (DynamoDbException e) {

```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("\nDone!");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeTable](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeTimeToLive

L'exemple de code suivant montre comment utiliser `DescribeTimeToLive`.

SDK pour Java 2.x

Décrivez la configuration TTL sur une table DynamoDB existante.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Optional;

    final DescribeTimeToLiveRequest request =
DescribeTimeToLiveRequest.builder()
        .tableName(tableName)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final DescribeTimeToLiveResponse response =
ddb.describeTimeToLive(request);
        System.out.println(tableName + " description of time to live is "
            + response.toString());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    }
```

```
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
    System.exit(0);
```

- Pour plus de détails sur l'API, reportez-vous [DescribeTimeToLive](#) à la section Référence des AWS SDK for Java 2.x API.

GetItem

L'exemple de code suivant montre comment utiliser `GetItem`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Récupère un élément d'une table à l'aide du `DynamoDbClient`.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Set;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
```



```
* its better practice to use the
* Enhanced Client, see the EnhancedGetItem example.
*/
public class GetItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyVal>

            Where:
                tableName - The Amazon DynamoDB table from which an item is
retrieved (for example, Music3).\s
                key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
                keyval - The key value that represents the item to get (for
example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Retrieving item \"%s\" from \"%s\"\\n", keyVal,
tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        getDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
            .s(keyVal)
            .build());
    }
}
```

```
GetItemRequest request = GetItemRequest.builder()
    .key(keyToGet)
    .tableName(tableName)
    .build();

try {
    // If there is no matching item, GetItem does not return any data.
    Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
    if (returnedItem.isEmpty())
        System.out.format("No item found with the key %s!\n", key);
    else {
        Set<String> keys = returnedItem.keySet();
        System.out.println("Amazon DynamoDB table attributes: \n");
        for (String key1 : keys) {
            System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
        }
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetItem](#) à la section Référence des AWS SDK for Java 2.x API.

ListTables

L'exemple de code suivant montre comment utiliser `ListTables`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                        .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
            }
        }
    }
}
```

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

        lastName = response.lastEvaluatedTableName();
        if (lastName == null) {
            moreTables = false;
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTables](#) à la section Référence des AWS SDK for Java 2.x API.

PutItem

L'exemple de code suivant montre comment utiliser `PutItem`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Met un élément dans un tableau à l'aide de [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
```

```

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedPutItem example.
 */
public class PutItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
<awardsval> <Songtitle> <songtitleval>

                Where:
                tableName - The Amazon DynamoDB table in which an item is placed
(for example, Music3).
                key - The key used in the Amazon DynamoDB table (for example,
Artist).
                keyval - The key value that represents the item to get (for
example, Famous Band).
                albumTitle - The Album title (for example, AlbumTitle).
                AlbumTitleValue - The name of the album (for example, Songs
About Life ).
                Awards - The awards column (for example, Awards).
                AwardVal - The value of the awards (for example, 10).
                SongTitle - The song title (for example, SongTitle).
                SongTitleVal - The value of the song title (for example, Happy
Day).

```

```
        **Warning** This program will place an item that you specify into a
table!
        """;

    if (args.length != 9) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String key = args[1];
    String keyVal = args[2];
    String albumTitle = args[3];
    String albumTitleValue = args[4];
    String awards = args[5];
    String awardVal = args[6];
    String songTitle = args[7];
    String songTitleVal = args[8];

    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
        songTitleVal);
    System.out.println("Done!");
    ddb.close();
}

public static void putItemInTable(DynamoDbClient ddb,
    String tableName,
    String key,
    String keyVal,
    String albumTitle,
    String albumTitleValue,
    String awards,
    String awardVal,
    String songTitle,
    String songTitleVal) {

    HashMap<String, AttributeValue> itemValues = new HashMap<>();
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
}
```

```
        itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
        itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item(itemValues)
                .build();

        try {
            PutItemResponse response = ddb.putItem(request);
            System.out.println(tableName + " was successfully updated. The request
id is "
                + response.responseMetadata().requestId());

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

- Pour plus de détails sur l'API, reportez-vous [PutItem](#) à la section Référence des AWS SDK for Java 2.x API.

Query

L'exemple de code suivant montre comment utiliser `Query`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Interroge une table à l'aide de [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedQueryRecords example.
 */
public class Query {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <partitionKeyName> <partitionKeyVal>

                Where:
                tableName - The Amazon DynamoDB table to put the item in (for
                example, Music3).
                partitionKeyName - The partition key name of the Amazon DynamoDB
                table (for example, Artist).
```



```
        partitionKeyVal - The value of the partition key that should
match (for example, Famous Band).
        """";

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String partitionKeyName = args[1];
    String partitionKeyVal = args[2];

    // For more information about an alias, see:
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeName.html
    String partitionAlias = "#a";

    System.out.format("Querying %s", tableName);
    System.out.println("");
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
partitionAlias);
    System.out.println("There were " + count + " record(s) returned");
    ddb.close();
}

public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal,
    String partitionAlias) {
    // Set up an alias for the partition key name in case it's a reserved word.
    HashMap<String, String> attrNameAlias = new HashMap<String, String>();
    attrNameAlias.put(partitionAlias, partitionKeyName);

    // Set up mapping of the partition name with the value.
    HashMap<String, AttributeValue> attrValues = new HashMap<>();
    attrValues.put(":" + partitionKeyName, AttributeValue.builder()
        .s(partitionKeyVal)
        .build());
}
```

```
    QueryRequest queryReq = QueryRequest.builder()
        .tableName(tableName)
        .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
        .expressionAttributeNames(attrNameAlias)
        .expressionAttributeValues(attrValues)
        .build();

    try {
        QueryResponse response = ddb.query(queryReq);
        return response.count();
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return -1;
}
}
```

Interroge une table à l'aide de `DynamoDbClient` et d'un index secondaire.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
 * year-index. For more information, see:
```

```
*
* https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
*/
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        queryIndex(ddb, tableName);
        ddb.close();
    }

    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(":yearValue",
AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                .tableName(tableName)
                .indexName("year-index")
                .keyConditionExpression("#year = :yearValue")
                .expressionAttributeNames(expressionAttributesNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus d'informations sur l'API, consultez [Requête](#) dans la référence d'API AWS SDK for Java 2.x .

Scan

L'exemple de code suivant montre comment utiliser Scan.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Analyse une table Amazon DynamoDB à l'aide de. [DynamoDbClient](#)

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */

public class DynamoDBScanItems {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <tableName>

    Where:
        tableName - The Amazon DynamoDB table to get information from
(for example, Music3).
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String tableName = args[0];
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

scanItems(ddb, tableName);
ddb.close();
}

public static void scanItems(DynamoDbClient ddb, String tableName) {
    try {
        ScanRequest scanRequest = ScanRequest.builder()
            .tableName(tableName)
            .build();

        ScanResponse response = ddb.scan(scanRequest);
        for (Map<String, AttributeValue> item : response.items()) {
            Set<String> keys = item.keySet();
            for (String key : keys) {
                System.out.println("The key name is " + key + "\n");
                System.out.println("The value is " + item.get(key).s());
            }
        }
    }

    } catch (DynamoDbException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
```

```
    }  
}
```

- Pour de plus amples informations sur API, consultez [TagResource](#) dans AWS SDK for Java 2.x Référence de l'API.

UpdateItem

L'exemple de code suivant montre comment utiliser `UpdateItem`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Met à jour un élément d'un tableau à l'aide de [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;  
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import java.util.HashMap;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better  
 * practice to use the  
 * Enhanced Client, See the EnhancedModifyItem example.  
 */
```

```
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyVal> <name> <updateVal>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music3).
                key - The name of the key in the table (for example, Artist).
                keyVal - The value of the key (for example, Famous Band).
                name - The name of the column where the value is updated (for
example, Awards).
                updateVal - The value used to update an item (for example, 14).
            Example:
                UpdateItem Music3 Artist Famous Band Awards 14
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String name = args[3];
        String updateVal = args[4];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
        ddb.close();
    }

    public static void updateTableItem(DynamoDbClient ddb,
        String tableName,
        String key,
        String keyVal,
        String name,
        String updateVal) {
```

```
HashMap<String, AttributeValue> itemKey = new HashMap<>();
itemKey.put(key, AttributeValue.builder()
    .s(keyVal)
    .build());

HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
updatedValues.put(name, AttributeValueUpdate.builder()
    .value(AttributeValue.builder().s(updateVal).build())
    .action(AttributeAction.PUT)
    .build());

UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(itemKey)
    .attributeUpdates(updatedValues)
    .build();

try {
    ddb.updateItem(request);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("The Amazon DynamoDB table was updated!");
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateItem](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateTimeToLive

L'exemple de code suivant montre comment utiliser `UpdateTimeToLive`.

SDK pour Java 2.x

Activez le TTL sur une table DynamoDB existante.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
```



```

import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
    .attributeName(ttlAttributeName)
    .enabled(true)
    .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
    .tableName(tableName)
    .timeToLiveSpecification(ttlSpecification)
    .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
        final UpdateTimeToLiveResponse response =
ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL successfully updated.
The request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");

```

Désactivez le TTL sur une table DynamoDB existante.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

```

```

import java.util.Optional;

    final Region region = Optional.ofNullable(args[2]).isEmpty() ?
Region.US_EAST_1 : Region.of(args[2]);
    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
        .attributeName(ttlAttributeName)
        .enabled(false)
        .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
        .tableName(tableName)
        .timeToLiveSpecification(ttlSpecification)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateTimeToLiveResponse response = ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL successfully updated. The
request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");

```

- Pour plus de détails sur l'API, reportez-vous [UpdateTimeToLive](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Mettre à jour le TTL d'un élément de manière conditionnelle

L'exemple de code suivant montre comment mettre à jour de manière conditionnelle le TTL d'un élément.

SDK pour Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

public class UpdateTTLConditional {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <newTtlAttribute> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
                sortKey - The name of the sort key. Also known as the range
attribute.
                newTtlAttribute - New attribute name (as part of the update
command)
                region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
            """;
        // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
        if (!(args.length == 4 || args.length == 5)) {
            System.out.println(usage);
            System.exit(1);
        }
        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final String newTtlAttribute = args[3];
```

```

    Region region = Optional.ofNullable(args[4]).isEmpty() ? Region.US_EAST_1 :
Region.of(args[4]);

    // Get current time in epoch second format
    final long currentTime = System.currentTimeMillis() / 1000;
    // Calculate expiration time 90 days from now in epoch second format
    final long expireDate = currentTime + (90 * 24 * 60 * 60);
    // An expression that defines one or more attributes to be updated, the
action to be performed on them, and new values for them.
    final String updateExpression = "SET newTtlAttribute = :val1";
    // A condition that must be satisfied in order for a conditional update to
succeed.
    final String conditionExpression = "expireAt > :val2";

    final ImmutableMap<String, AttributeValue> keyMap =
        ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
            "sortKey", AttributeValue.fromS(sortKey));
    final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
        ":val1", AttributeValue.builder().s(newTtlAttribute).build(),
        ":val2",
        AttributeValue.builder().s(String.valueOf(expireDate)).build()
    );

    final UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(keyMap)
        .updateExpression(updateExpression)
        .conditionExpression(conditionExpression)
        .expressionAttributeValues(expressionAttributeValues)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateItemResponse response = ddb.updateItem(request);
        System.out.println(tableName + " UpdateItem operation with conditional
TTL successful. Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
    }

```

```
        System.exit(1);
    }
    System.exit(0);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateItem](#) à la section Référence des AWS SDK for Java 2.x API.

Création d'un article avec un TTL

L'exemple de code suivant montre comment créer un élément avec TTL.

SDK pour Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.io.Serializable;
import java.util.Map;
import java.util.Optional;

public class CreateTTL {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
                or partition key.
```

```

        sortKey - The name of the sort key. Also known as the range
attribute.
        region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
        """;
    // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
    if (!(args.length == 3 || args.length == 4)) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String primaryKey = args[1];
    String sortKey = args[2];
    Region region = Optional.ofNullable(args[3]).isEmpty() ? Region.US_EAST_1 :
Region.of(args[3]);

    // Get current time in epoch second format
    final long createDate = System.currentTimeMillis() / 1000;

    // Calculate expiration time 90 days from now in epoch second format
    final long expireDate = createDate + (90 * 24 * 60 * 60);

    final ImmutableMap<String, ? extends Serializable> itemMap =
        ImmutableMap.of("primaryKey", primaryKey,
            "sortKey", sortKey,
            "creationDate", createDate,
            "expireAt", expireDate);
    final PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item((Map<String, AttributeValue>) itemMap)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " PutItem operation with TTL successful.
Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    }

```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.exit(0);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutItem](#) à la section Référence des AWS SDK for Java 2.x API.

Commencer à utiliser des tables, des éléments et des requêtes

L'exemple de code suivant illustre comment :

- Créez une table pouvant contenir des données vidéo.
- Insérer, récupérez et mettez à jour un seul film dans la table.
- Écrivez des données vidéo dans la table à partir d'un exemple de fichier JSON.
- Recherchez les films sortis au cours d'une année donnée.
- Recherchez les films sortis au cours d'une plage d'années spécifique.
- Supprimez un film de la table, puis supprimez la table.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez une table DynamoDB.

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
```

```
attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("year")
    .attributeType("N")
    .build());

attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("title")
    .attributeType("S")
    .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
```



```

        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

Créez une fonction d'assistance pour télécharger et extraire l'exemple de fichier JSON.

```

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);
    }
}

```

```
        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}
```

Obtenez un élément d'une table.

```
public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

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

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

Exemple complet.

```
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * This Java example performs these tasks:  
 *  
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.  
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the  
 * Enhanced client.  
 * 3. Gets data from the Movie table.  
 * 4. Adds a new item.  
 * 5. Updates an item.  
 * 6. Uses a Scan to query items using the Enhanced client.  
 * 7. Queries all items where the year is 2013 using the Enhanced Client.  
 * 8. Deletes the table.  
 */  
  
public class Scenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
  
    public static void main(String[] args) throws IOException {  
        final String usage = ""  
  
            Usage:  
            <fileName>  
  
            Where:  
            fileName - The path to the moviedata.json file that you can  
download from the Amazon DynamoDB Developer Guide.  
            "";  
  
        if (args.length != 1) {  
            System.out.println(usage);  
        }  
    }  
}
```

```
        System.exit(1);
    }

    String tableName = "Movies";
    String fileName = args[0];
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon DynamoDB example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(
        "1. Creating an Amazon DynamoDB table named Movies with a key named
year and a sort key named title.");
    createTable(ddb, tableName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Loading data into the Amazon DynamoDB table.");
    loadData(ddb, tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Getting data from the Movie table.");
    getItem(ddb);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Putting a record into the Amazon DynamoDB table.");
    putRecord(ddb);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Updating a record.");
    updateTableItem(ddb, tableName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Scanning the Amazon DynamoDB table.");
    scanMovies(ddb, tableName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
```

```
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " + newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder()
                .partitionValue(2013)
                .build());
```

```
        // Get items in the table and write out the ID value.
        Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
        String result = "";

        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The title of the movie is " + rec.getTitle());
            System.out.println("The movie information is " + rec.getInfo());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {
    System.out.println("***** Scanning all movies.\n");
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        Iterator<Movies> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The movie title is " + rec.getTitle());
            System.out.println("The movie year is " + rec.getYear());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
```

```

        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName) {
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s("{\"directors\": [\"Merian C.
Cooper\", \"Ernest B. Schoedsack\"]")
            .build())

```



```
        .action(AttributeAction.PUT)
        .build());

UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(itemKey)
    .attributeUpdates(updatedValues)
    .build();

try {
    ddb.updateItem(request);
} catch (ResourceNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

System.out.println("Item was updated!");
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

public static void putRecord(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();
    }
}
```

```
DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

// Populate the Table.
Movies record = new Movies();
record.setYear(2020);
record.setTitle("My Movie2");
record.setInfo("no info");
table.putItem(record);

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Added a new movie to the table.");
}

public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

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

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        }
    }
}
```

```
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [Interrogation](#)
 - [Analyser](#)
 - [UpdateItem](#)

Interroger une table à l'aide de lots d'instructions PartiQL

L'exemple de code suivant illustre comment :

- Obtenez un lot d'éléments en exécutant plusieurs instructions SELECT.
- Ajoutez un lot d'éléments en exécutant plusieurs instructions INSERT.
- Mettez à jour un lot d'éléments en exécutant plusieurs instructions UPDATE.
- Supprimez un lot d'éléments en exécutant plusieurs instructions DELETE.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("***** Creating an Amazon DynamoDB table named
" + tableName
            + " with a key named year and a sort key named
title.");
        createTable(ddb, tableName);

        System.out.println("***** Adding multiple records into the " +
tableName
            + " table using a batch command.");
        putRecordBatch(ddb);

        System.out.println("***** Updating multiple records using a batch
command.");
        updateTableItemBatch(ddb);

        System.out.println("***** Deleting multiple records using a batch
command.");
        deleteItemBatch(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
```

```
        ArrayList<AttributeDefinition> attributeDefinitions = new
ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)

        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest =
DescribeTableRequest.builder()
```

```
        .tableName(tableName)
        .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter
        .waitUntilTableExists(tableRequest);

waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully
created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE
{'year':?, 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie 1")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);
```

```
BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parameters)
    .build();

// Set data for Movie 2.
List<AttributeValue> parametersMovie2 = new ArrayList<>();
AttributeValue attMovie2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie2A = AttributeValue.builder()
    .s("My Movie 2")
    .build();

AttributeValue attMovie2B = AttributeValue.builder()
    .s("No Information")
    .build();

parametersMovie2.add(attMovie2);
parametersMovie2.add(attMovie2A);
parametersMovie2.add(attMovie2B);

BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersMovie2)
    .build();

// Set data for Movie 3.
List<AttributeValue> parametersMovie3 = new ArrayList<>();
AttributeValue attMovie3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie3A = AttributeValue.builder()
    .s("My Movie 3")
    .build();

AttributeValue attMovie3B = AttributeValue.builder()
    .s("No Information")
    .build();
```

```

        parametersMovie3.add(attMovie3);
        parametersMovie3.add(attMovie3A);
        parametersMovie3.add(attMovie3B);

        BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
                        .statement(sqlStatement)
                        .parameters(parametersMovie3)
                        .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();

        myBatchStatementList.add(statementRequestMovie1);
        myBatchStatementList.add(statementRequestMovie2);
        myBatchStatementList.add(statementRequestMovie3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                                .statements(myBatchStatementList)
                                .build();

        BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
        System.out.println("ExecuteStatement successful: " +
response.toString());
        System.out.println("Added new movies using a batch
command.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItemBatch(DynamoDbClient ddb) {
    String sqlStatement = "UPDATE MoviesPartiQBatch SET info =
'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack' where year=? and
title=?";

    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Update three records.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))

```



```
        .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Update record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Update record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
```

```
        parametersRec3.add(attRec3a);
        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch
command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ?
and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();
```

```
AttributeValue att2 = AttributeValue.builder()
    .s("My Movie 1")
    .build();

parametersRec1.add(att1);
parametersRec1.add(att2);

BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec1)
    .build();

// Specify record 2.
List<AttributeValue> parametersRec2 = new ArrayList<>();
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Specify record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
```

```
        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
                        .statement(sqlStatement)
                        .parameters(parametersRec3)
                        .build();

        // Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                        .statements(myBatchStatementList)
                        .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch
command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
    {
        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
}
```

```
private static ExecuteStatementResponse
executeStatementRequest(DynamoDbClient ddb, String statement,
                        List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}
```

- Pour plus de détails sur l'API, reportez-vous [BatchExecuteStatement](#) à la section Référence des AWS SDK for Java 2.x API.

Interroger une table à l'aide de PartiQL

L'exemple de code suivant illustre comment :

- Obtenez un élément en exécutant une instruction SELECT.
- Ajoutez un élément en exécutant une instruction INSERT.
- Mettez à jour un élément en exécutant une instruction UPDATE.
- Supprimez un élément en exécutant une instruction DELETE.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        final String usage = ""
```

Usage:

```
        <fileName>

        Where:
            fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String fileName = args[0];
    String tableName = "MoviesPartiQ";
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    System.out.println(
        "***** Creating an Amazon DynamoDB table named MoviesPartiQ with a
key named year and a sort key named title.");
    createTable(ddb, tableName);

    System.out.println("***** Loading data into the MoviesPartiQ table.");
    loadData(ddb, fileName);

    System.out.println("***** Getting data from the MoviesPartiQ table.");
    getItem(ddb);

    System.out.println("***** Putting a record into the MoviesPartiQ table.");
    putRecord(ddb);

    System.out.println("***** Updating a record.");
    updateTableItem(ddb);

    System.out.println("***** Querying the movies released in 2013.");
    queryTable(ddb);

    System.out.println("***** Deleting the Amazon DynamoDB table.");
    deleteDynamoDBTable(ddb, tableName);
    ddb.close();
}
```

```
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
```

```

        .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    List<AttributeValue> parameters = new ArrayList<>();
    while (iter.hasNext()) {

        // Add 200 movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf(year))
            .build();

        AttributeValue att2 = AttributeValue.builder()

```



```
        .s(title)
        .build();

    AttributeValue att3 = AttributeValue.builder()
        .s(info)
        .build();

    parameters.add(att1);
    parameters.add(att2);
    parameters.add(att3);

    // Insert the movie into the Amazon DynamoDB table.
    executeStatementRequest(ddb, sqlStatement, parameters);
    System.out.println("Added Movie " + title);

    parameters.remove(att1);
    parameters.remove(att2);
    parameters.remove(att3);
    t++;
}
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The Perks of Being a Wallflower")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());
    } catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void putRecord(DynamoDbClient ddb) {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    try {
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2020"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added new movie.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItem(DynamoDbClient ddb) {

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
```

```
        .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The East")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
```

```
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement,
        List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }

    private static void processResults(ExecuteStatementResponse
executeStatementResult) {
        System.out.println("ExecuteStatement successful: " +
executeStatementResult.toString());
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ExecuteStatement](#) à la section Référence des AWS SDK for Java 2.x API.

Requête d'éléments TTL

L'exemple de code suivant montre comment rechercher des éléments TTL.

SDK pour Java 2.x

Expression filtrée par requête pour rassembler des éléments TTL dans une table DynamoDB.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

// Get current time in epoch second format (comparing against expiry
attribute)
final long currentTime = System.currentTimeMillis() / 1000;

// A string that contains conditions that DynamoDB applies after the Query
operation, but before the data is returned to you.
final String keyConditionExpression = "#pk = :pk";

// The condition that specifies the key values for items to be retrieved by
the Query action.
final String filterExpression = "#ea > :ea";
final Map<String, String> expressionAttributeNames = ImmutableMap.of(
    "#pk", "primaryKey",
    "#ea", "expireAt");
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":pk", AttributeValue.builder().s(primaryKey).build(),
    ":ea",
AttributeValue.builder().s(String.valueOf(currentTime)).build()
);

final QueryRequest request = QueryRequest.builder()
    .tableName(tableName)
    .keyConditionExpression(keyConditionExpression)
    .filterExpression(filterExpression)
    .expressionAttributeNames(expressionAttributeNames)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
```

```

try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final QueryResponse response = ddb.query(request);
    System.out.println(tableName + " Query operation with TTL successful.
Request id is "
        + response.responseMetadata().requestId());
    // Print the items that are not expired
    for (Map<String, AttributeValue> item : response.items()) {
        System.out.println(item.toString());
    }
} catch (ResourceNotFoundException e) {
    System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.exit(0);

```

- Pour plus d'informations sur l'API, consultez [Requête](#) dans la référence d'API AWS SDK for Java 2.x .

Mettre à jour le TTL d'un article

L'exemple de code suivant montre comment mettre à jour le TTL d'un élément.

SDK pour Java 2.x

Mettez à jour le TTL sur un élément DynamoDB existant dans une table.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;

```

```
import java.util.Optional;

// Get current time in epoch second format
final long currentTime = System.currentTimeMillis() / 1000;
// Calculate expiration time 90 days from now in epoch second format
final long expireDate = currentTime + (90 * 24 * 60 * 60);
// An expression that defines one or more attributes to be updated, the
action to be performed on them, and new values for them.
final String updateExpression = "SET updatedAt=:c, expireAt=:e";

final ImmutableMap<String, AttributeValue> keyMap =
    ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
        "sortKey", AttributeValue.fromS(sortKey));
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":c",
AttributeValue.builder().s(String.valueOf(currentTime)).build(),
    ":e", AttributeValue.builder().s(String.valueOf(expireDate)).build()
);

final UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(keyMap)
    .updateExpression(updateExpression)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final UpdateItemResponse response = ddb.updateItem(request);
    System.out.println(tableName + " UpdateItem operation with TTL
successful. Request id is "
        + response.responseMetadata().requestId());
} catch (ResourceNotFoundException e) {
    System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.exit(0);
```

- Pour plus de détails sur l'API, reportez-vous [UpdateItem](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples sans serveur

Signalement des défaillances d'éléments de lot pour les fonctions Lambda à l'aide d'un déclencheur DynamoDB

L'exemple de code suivant montre comment implémenter une réponse par lots partielle pour les fonctions Lambda qui reçoivent des événements d'un flux DynamoDB. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Signaler les défaillances d'éléments de lot DynamoDB avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {
```



```
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
input.getRecords()) {
            try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse();
    }
}
```

Exemples d'Amazon EC2 utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon EC2.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Hello Amazon EC2

Les exemples de code suivants montrent comment démarrer avec Amazon EC2.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

AllocateAddress

L'exemple de code suivant montre comment utiliser `AllocateAddress`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
        AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
        ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [AllocateAddress](#) à la section Référence des AWS SDK for Java 2.x API.

AssociateAddress

L'exemple de code suivant montre comment utiliser `AssociateAddress`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [AssociateAddress](#) à la section Référence des AWS SDK for Java 2.x API.

AuthorizeSecurityGroupIngress

L'exemple de code suivant montre comment utiliser `AuthorizeSecurityGroupIngress`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
```

```
        .ipPermissions(ipPerm, ipPerm2)
        .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [AuthorizeSecurityGroupIngress](#) à la section Référence des AWS SDK for Java 2.x API.

CreateKeyPair

L'exemple de code suivant montre comment utiliser `CreateKeyPair`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
```

```
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyPair](#) à la section Référence des AWS SDK for Java 2.x API.

CreateSecurityGroup

L'exemple de code suivant montre comment utiliser `CreateSecurityGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
```

```
        .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteKeyPair

L'exemple de code suivant montre comment utiliser `DeleteKeyPair`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteSecurityGroup

L'exemple de code suivant montre comment utiliser `DeleteSecurityGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
```

```
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeInstanceTypes

L'exemple de code suivant montre comment utiliser `DescribeInstanceTypes`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();
```

```
DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
for (InstanceTypeInfo type : instanceTypes) {
    System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
    System.out.println("Network information is " +
type.networkInfo().toString());
    System.out.println("Instance type is " +
type.instanceType().toString());
    instanceType = type.instanceType().toString();
    if (instanceType.compareTo("t2.2xlarge") == 0){
        return instanceType;
    }
}

} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceTypes](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeInstances

L'exemple de code suivant montre comment utiliser `DescribeInstances`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
```

```
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
instance.instanceType());
                    System.out.println("Instance state name is " +
instance.state().name());
                    System.out.println("Monitoring information is " +
instance.monitoring().state());
                });
        }
    }
}
```

```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorCode());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeKeyPairs

L'exemple de code suivant montre comment utiliser `DescribeKeyPairs`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeSecurityGroups

L'exemple de code suivant montre comment utiliser `DescribeSecurityGroups`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section Référence des AWS SDK for Java 2.x API.

DisassociateAddress

L'exemple de code suivant montre comment utiliser `DisassociateAddress`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");


    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateAddress](#) à la section Référence des AWS SDK for Java 2.x API.

ReleaseAddress

L'exemple de code suivant montre comment utiliser `ReleaseAddress`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();


        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section Référence des AWS SDK for Java 2.x API.

RunInstances

L'exemple de code suivant montre comment utiliserRunInstances.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
```



```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This code example requires an AMI value. You can learn more about this value
 * by reading this documentation topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
 */
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <amiId>

            Where:
                name - An instance name value that you can obtain from the AWS
                Console (for example, ami-xxxxxx5c8b987b1a0).\s
                amiId - An Amazon Machine Image (AMI) value that you can obtain
                from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String amiId = args[1];
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
```

```
        .region(region)
        .build();

String instanceId = createEC2Instance(ec2, name, amiId);
System.out.println("The Amazon EC2 Instance ID is " + instanceId);
ec2.close();
}

public static String createEC2Instance(Ec2Client ec2, String name, String amiId)
{
    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    // Use a waiter to wait until the instance is running.
    System.out.println("Going to start an EC2 instance using a waiter");
    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceIdVal = response.instances().get(0).instanceId();
    ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceIdVal)
        .tags(tag)
        .build();

    try {
        ec2.createTags(tagRequest);
        System.out.printf("Successfully started EC2 Instance %s based on AMI
%s", instanceIdVal, amiId);
        return instanceIdVal;

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section Référence des AWS SDK for Java 2.x API.

StartInstances

L'exemple de code suivant montre comment utiliser `StartInstances`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void startInstance(Ec2Client ec2, String instanceId) {  
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()  
        .overrideConfiguration(b -> b.maxAttempts(100))  
        .client(ec2)  
        .build();  
  
    StartInstancesRequest request = StartInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    System.out.println("Use an Ec2Waiter to wait for the instance to run. This  
will take a few minutes.");  
    ec2.startInstances(request);  
    DescribeInstancesRequest instanceRequest =  
DescribeInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    WaiterResponse<DescribeInstancesResponse> waiterResponse =  
ec2Waiter.waitUntilInstanceRunning(instanceRequest);  
    waiterResponse.matched().response().ifPresent(System.out::println);  
    System.out.println("Successfully started instance " + instanceId);  
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section Référence des AWS SDK for Java 2.x API.

StopInstances

L'exemple de code suivant montre comment utiliser `StopInstances`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section Référence des AWS SDK for Java 2.x API.

TerminateInstances

L'exemple de code suivant montre comment utiliser `TerminateInstances`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Créer et gérer un service résilient

L'exemple de code suivant montre comment créer un service Web à charge équilibrée qui renvoie des recommandations de livres, de films et de chansons. L'exemple montre comment le service répond aux défaillances et comment le restructurer pour accroître la résilience en cas de défaillance.

- Utilisez un groupe Amazon EC2 Auto Scaling pour créer des instances Amazon Elastic Compute Cloud (Amazon EC2) sur la base d'un modèle de lancement et pour maintenir le nombre d'instances dans une plage spécifiée.
- Gérez et distribuez les requêtes HTTP avec Elastic Load Balancing.
- Surveillez l'état des instances d'un groupe Auto Scaling et transférez les demandes uniquement aux instances saines.
- Exécutez un serveur Web Python sur chaque instance EC2 pour gérer les requêtes HTTP. Le serveur Web répond par des recommandations et des surveillances de l'état.
- Simulez un service de recommandation avec une table Amazon DynamoDB.
- Contrôlez la réponse du serveur Web aux demandes et aux contrôles de santé en mettant à jour AWS Systems Manager les paramètres.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
public class Main {

    public static final String fileName = "C:\\\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
        System.out.println(DASHES);
    }
}
```

```
System.out.println(DASHES);
System.out.println("A - SETUP THE RESOURCES");
System.out.println("Press Enter when you're ready to start deploying
resources.");
in.nextLine();
deploy(loadBalancer);
System.out.println(DASHES);
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.

    To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
        """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
```



```

        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
        throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScalingGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);

```

```
System.out.println("Creating and populating a DynamoDB table named " +
tableName);
Database database = new Database();
database.createTable(tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
    This script starts a Python web server defined in the `server.py`
script. The web server
    listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
    For demo purposes, this server is run as the root user. In
production, the best practice is to
    run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to
assume a role that grants
    permissions to access the DynamoDB recommendation table and Systems
Manager parameters
    that control the flow of the demo.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
```

```
        HTTP requests. You can see these instances in the console or
continue with the demo.
        Press Enter when you're ready to continue.
        """);

    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating variables that control the flow of the demo.");
    ParameterHelper paramHelper = new ParameterHelper();
    paramHelper.reset();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an Elastic Load Balancing target group and load balancer.
The target group
        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
```

```

        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {

```

```
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        """);
}
```

```
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
            ""
                \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
            """);
        demoChoices(loadBalancer);

        System.out.println(
            ""
                Instead of failing when the recommendation service fails,
the web service can return a static response.
                While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
            """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
            Now, sending a GET request to the load balancer endpoint returns a
static response.
            The service still reports as healthy because health checks are still
shallow.
            """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
            Let's also substitute bad credentials for one of the instances in
the target group so that it can't
            access the DynamoDB recommendation table. We will get an instance id
value.
            """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();
```

```
// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
+ " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");
```

```
        System.out.println("""
            Now, checking target health indicates that the instance with bad
credentials
            is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
            instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
            the load balancer takes unhealthy instances out of its rotation.
            """);

        demoChoices(loadBalancer);

        System.out.println(
            """
                Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
                """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
```



```
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
```

```

        new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
check to update
                                Note that it can take a minute or two for the health
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {

```

```
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Créez une classe qui englobe les actions Auto Scaling et Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
```

```
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
    .build();

// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    }
}
```

```

    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.getInstanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
    .instanceIds(instanceId)
    .documentName("AWS-RunShellScript")
    .parameters(Collections.singletonMap("commands",
        Collections.singletonList("cd / && sudo python3 server.py
80")))
    .build();

getSSMClient().sendCommand(sendCommandRequest);
System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress) {
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,

```

```
    * and deletes all the resources.
    */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                .builder()
                .instanceProfileName(profileName)
                .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .roleName(roleName)
                .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(attachedPolicy.policyArn())
```

```

        .build();

        getIAMClient().detachRolePolicy(request);
        System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
    }

    getIAMClient().deleteRole(deleteRoleRequest);
    System.out.println("Instance profile and role deleted.");

} catch (IamException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 */

```



```
*
*/
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                }
            }
        }
    }
}
```

```

        portIsOpen = true;
    }

    if (!portIsOpen) {
        System.out
            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
    } else {
        break;
    }
}
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);
    }
}

```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

    String availabilityZones = String.join(",", availabilityZoneNames);
    LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
        .launchTemplateName(templateName)
        .version("$Default")
        .build();

    String[] zones = availabilityZones.split(",");
    CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
        .launchTemplate(specification)
        .availabilityZones(zones)
        .maxSize(groupSize)
        .minSize(groupSize)
        .autoScalingGroupName(autoScalingGroupName)
        .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);
    }
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();
```

```
DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
    .filters(vpcFilter, azFilter, defaultForAZ)
    .build();

DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
subnets = response.subnets();
return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();
```

```

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
            .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
            .builder()
            .policyArn(policy.arn())
            .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                    .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                    || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .roleName(roleName) // Specify the name of the IAM role
                        .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();
            }
        }
    }

```

```
        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
System.out.println(InstanceProfile + " Deleted");
System.out.println("All roles and policies are deleted.");
}
}
```

Créez une classe qui englobe les actions Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
        DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
        getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
        DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
        getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```



```

    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {

```

```
boolean success = false;
int retries = 3;
CloseableHttpClient httpClient = HttpClients.createDefault();

// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/**
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
```

```
        .name(targetGroupName)
        .protocol(protocol)
        .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Créez une classe qui utilise DynamoDB pour simuler un service de recommandation.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /**
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```

```
* MediaType,
* forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();
    }
}
```

```
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
}
```

```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Créez une classe qui englobe les actions de Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)

- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Démarrer avec les instances

L'exemple de code suivant illustre comment :

- Créez une paire de clés et un groupe de sécurité.
- Sélectionnez une Amazon Machine Image (AMI) et un type d'instance compatible, puis créez une instance.

• Arrêtez l'instance, puis redémarrez-la.

- Associez une adresse IP Elastic à votre instance
- Connectez-vous à votre instance avec SSH, puis nettoyez les ressources.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets more information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI's
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
```

```

public class EC2Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {

        final String usage = ""

            Usage:
                <keyName> <fileName> <groupName> <groupDesc> <vpcId>

            Where:
                keyName - A key pair name (for example, TestKeyPair).\s
                fileName - A file name where the key information is written to.
\s
                groupName - The name of the security group.\s
                groupDesc - The description of the security group.\s
                vpcId - A VPC Id value. You can get this value from the AWS
Management Console.\s
                myIpAddress - The IP address of your development machine.\s

            """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyName = args[0];
        String fileName = args[1];
        String groupName = args[2];
        String groupDesc = args[3];
        String vpcId = args[4];
        String myIpAddress = args[5];

        Region region = Region.US_WEST_2;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
    }
}

```

```
System.out.println("Welcome to the Amazon EC2 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
createKeyPair(ec2, keyName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List key pairs.");
describeKeys(ec2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create a security group.");
String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Display security group info for the newly created
security group.");
describeSecurityGroups(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
String instanceId = getParaValues(ssmClient);
System.out.println("The instance Id is " + instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get more information about an amzn2 image.");
String amiValue = describeImage(ec2, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of instance types.");
String instanceType = getInstanceTypes(ec2);
System.out.println("The instance type is " + instanceType);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("8. Create an instance.");
String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue);
System.out.println("The instance Id is " + newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance. ");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Allocate an Elastic IP address and associate it with
the instance.");
String allocationId = allocateAddress(ec2);
System.out.println("The allocation Id value is " + allocationId);
String associationId = associateAddress(ec2, newInstanceId, allocationId);
System.out.println("The associate Id value is " + associationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Describe the instance again.");
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("14. Disassociate and release the Elastic IP address.");
disassociateAddress(ec2, associationId);
releaseEC2Address(ec2, allocationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Terminate the instance and use a waiter.");
terminateEC2(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the security group.");
deleteEC2SecGroup(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Delete the key.");
deleteKeys(ec2, keyName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("You successfully completed the Amazon EC2 scenario.");
System.out.println(DASHES);
ec2.close();
}

public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateEC2(Ec2Client ec2, String instanceId) {
```

```
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();
    }
}
```



```
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();
```

```
    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitForInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
}

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitForInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}

public static String describeEC2Instances(Ec2Client ec2, String newInstanceId) {
    try {
        String pubAddress = "";
        boolean isRunning = false;
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        while (!isRunning) {
            DescribeInstancesResponse response = ec2.describeInstances(request);
            String state =
response.reservations().get(0).instances().get(0).state().name().name();
            if (state.compareTo("RUNNING") == 0) {
                System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
            }
        }
    }
}
```

```

        System.out.println(
            "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
        System.out.println(
            "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
        pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
        System.out.println("Instance address is " + pubAddress);
        isRunning = true;
    }
}
return pubAddress;
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
    String amiId) {
    try {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r ->
response.reservations().get(0).instances().get(0).state().name());
        System.out.println("Successfully started EC2 instance " + instanceIdVal
+ " based on AMI " + amiId);
        return instanceIdVal;

    } catch (SsmException e) {
        System.err.println(e.getMessage());
    }
}

```

```
        System.exit(1);
    }
    return "";
}

// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.getInstanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());
            System.out.println("Instance type is " +
type.getInstanceType().toString());
            instanceType = type.getInstanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(instanceId)
            .build();
```

```
        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is " +
response.images().get(0).description());
        System.out.println("The name of the first image is " +
response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for
(ssoftware.amazon.awssdk.services.ssm.model.GetParametersByPathResponse response :
responses) {
            System.out.println("Test " + response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para : parameterList) {
                System.out.println("The name of the para is: " + para.name());
                System.out.println("The type of the para is: " + para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }
    }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
        return "";
    }

    // Return true if the name has amzn2 in it. For example:
    // /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
    private static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            // Use a paginator.
            DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
            listGroups.stream()
                .flatMap(r -> r.securityGroups().stream())
                .forEach(group -> System.out
                    .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
        String myIpAddress) {
        try {
            CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
                .groupName(groupName)
                .description(groupDesc)
                .vpcId(vpcId)
                .build();
```

```
        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
```

```
        keyPair.keyName(),
        keyPair.keyFingerprint()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)

- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

Exemples d'Amazon ECS utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon ECS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateCluster

L'exemple de code suivant montre comment utiliser `CreateCluster`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterName>\s

                Where:
                clusterName - The name of the ECS cluster to create.
                """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String clusterName = args[0];
    Region region = Region.US_EAST_1;
    EcsClient ecsClient = EcsClient.builder()
        .region(region)
        .build();

    String clusterArn = createGivenCluster(ecsClient, clusterName);
    System.out.println("The cluster ARN is " + clusterArn);
    ecsClient.close();
}

public static String createGivenCluster(EcsClient ecsClient, String clusterName)
{
    try {
        ExecuteCommandConfiguration commandConfiguration =
ExecuteCommandConfiguration.builder()
            .logging(ExecuteCommandLogging.DEFAULT)
            .build();

        ClusterConfiguration clusterConfiguration =
ClusterConfiguration.builder()
            .executeCommandConfiguration(commandConfiguration)
            .build();

        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterName(clusterName)
            .configuration(clusterConfiguration)
            .build();

        CreateClusterResponse response =
ecsClient.createCluster(clusterRequest);
        return response.cluster().clusterArn();

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCluster](#) à la section Référence des AWS SDK for Java 2.x API.

CreateService

L'exemple de code suivant montre comment utiliser `CreateService`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateService {
    public static void main(String[] args) {
        final String usage = ""

                                Usage:
                                <clusterName> <serviceName> <securityGroups>
                                <subnets> <taskDefinition>
```

```

        create.

        Where:
            clusterName - The name of the ECS cluster.
            serviceName - The name of the ECS service to

            securityGroups - The name of the security group.
            subnets - The name of the subnet.
            taskDefinition - The name of the task definition.
        """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceName = args[1];
        String securityGroups = args[2];
        String subnets = args[3];
        String taskDefinition = args[4];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        String serviceArn = createNewService(ecsClient, clusterName,
serviceName, securityGroups, subnets,
            taskDefinition);
        System.out.println("The ARN of the service is " + serviceArn);
        ecsClient.close();
    }

    public static String createNewService(EcsClient ecsClient,
        String clusterName,
        String serviceName,
        String securityGroups,
        String subnets,
        String taskDefinition) {

        try {
            AwsVpcConfiguration vpcConfiguration =
            AwsVpcConfiguration.builder()
                .securityGroups(securityGroups)
                .subnets(subnets)

```

```
        .build();

        NetworkConfiguration configuration =
NetworkConfiguration.builder()
                        .awsVpcConfiguration(vpcConfiguration)
                        .build();

        CreateServiceRequest serviceRequest =
CreateServiceRequest.builder()
                        .cluster(clusterName)
                        .networkConfiguration(configuration)
                        .desiredCount(1)
                        .launchType(LaunchType.FARGATE)
                        .serviceName(serviceName)
                        .taskDefinition(taskDefinition)
                        .build();

        CreateServiceResponse response =
ecsClient.createService(serviceRequest);
        return response.service().serviceArn();


    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateService](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteService

L'exemple de code suivant montre comment utiliser `DeleteService`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteService {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clusterName> <serviceArn>\s

            Where:
                clusterName - The name of the ECS cluster.
                serviceArn - The ARN of the ECS service.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
```

```
EcsClient ecsClient = EcsClient.builder()
    .region(region)
    .build();

deleteSpecificService(ecsClient, clusterName, serviceArn);
ecsClient.close();
}

public static void deleteSpecificService(EcsClient ecsClient, String
clusterName, String serviceArn) {
    try {
        DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
            .cluster(clusterName)
            .service(serviceArn)
            .build();

        ecsClient.deleteService(serviceRequest);
        System.out.println("The Service was successfully deleted");

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteService](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeClusters

L'exemple de code suivant montre comment utiliser `DescribeClusters`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterArn> \s

                Where:
                clusterArn - The ARN of the ECS cluster to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
```

```
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
        .clusters(clusterArn)
        .build();

        DescribeClustersResponse response =
ecsClient.describeClusters(clustersRequest);
        List<Cluster> clusters = response.clusters();
        for (Cluster cluster : clusters) {
            System.out.println("The cluster name is " + cluster.clusterName());
        }

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeClusters](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeTasks

L'exemple de code suivant montre comment utiliser `DescribeTasks`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterArn> <taskId>\s

                Where:
                clusterArn - The ARN of an ECS cluster.
                taskId - The task Id value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        String taskId = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        getAllTasks(ecsClient, clusterArn, taskId);
        ecsClient.close();
    }

    public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
taskId) {
        try {
            DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
                .cluster(clusterArn)
                .tasks(taskId)
```

```
        .build();

        DescribeTasksResponse response = ecsClient.describeTasks(tasksRequest);
        List<Task> tasks = response.tasks();
        for (Task task : tasks) {
            System.out.println("The task ARN is " + task.taskDefinitionArn());
        }

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeTasks](#) à la section Référence des AWS SDK for Java 2.x API.

ListClusters

L'exemple de code suivant montre comment utiliser `ListClusters`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        listAllClusters(ecsClient);
        ecsClient.close();
    }

    public static void listAllClusters(EcsClient ecsClient) {
        try {
            ListClustersResponse response = ecsClient.listClusters();
            List<String> clusters = response.clusterArns();
            for (String cluster : clusters) {
                System.out.println("The cluster arn is " + cluster);
            }
        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListClusters](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateService

L'exemple de code suivant montre comment utiliser UpdateService.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class UpdateService {

    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <clusterName> <serviceArn>\s

            Where:
                clusterName - The cluster name.
                serviceArn - The service ARN value.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
```

```
String serviceArn = args[1];
Region region = Region.US_EAST_1;
EcsClient ecsClient = EcsClient.builder()
    .region(region)
    .build();

updateSpecificService(ecsClient, clusterName, serviceArn);
ecsClient.close();
}

public static void updateSpecificService(EcsClient ecsClient, String
clusterName, String serviceArn) {
    try {
        UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
            .cluster(clusterName)
            .service(serviceArn)
            .desiredCount(0)
            .build();

        ecsClient.updateService(serviceRequest);
        System.out.println("The service was modified");

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateService](#) à la section Référence des AWS SDK for Java 2.x API.

Elastic Load Balancing - Exemples de version 2 utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de la AWS SDK for Java 2.x version 2 d'Elastic Load Balancing.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Elastic Load Balancing

Les exemples de code suivants montrent comment démarrer avec Elastic Load Balancing.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public class HelloLoadBalancer {

    public static void main(String[] args) {
        ElasticLoadBalancingV2Client loadBalancingV2Client =
ElasticLoadBalancingV2Client.builder()
                                .region(Region.US_EAST_1)
                                .build();

        DescribeLoadBalancersResponse loadBalancersResponse =
loadBalancingV2Client
                                .describeLoadBalancers(r -> r.pageSize(10));
        List<LoadBalancer> loadBalancerList =
loadBalancersResponse.loadBalancers();
        for (LoadBalancer lb : loadBalancerList)
            System.out.println("Load Balancer DNS name = " +
lb.dnsName());
    }
}
```


- Pour plus de détails sur l'API, reportez-vous [DescribeLoadBalancers](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateListener

L'exemple de code suivant montre comment utiliser `CreateListener`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();
```

```
// Create and wait for the load balancer to become available.
CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
    .loadBalancerArns(lbARN)
    .build();

System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
    .waitUntilLoadBalancerAvailable(request);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Load Balancer " + lbName + " is available.");

// Get the DNS name (endpoint) of the load balancer.
String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

// Create a listener for the load balance.
Action action = Action.builder()
    .targetGroupArn(targetGroupARN)
    .type("forward")
    .build();

CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

getLoadBalancerClient().createListener(listenerRequest);
System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
    + targetGroupARN);
```

```
        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateListener](#) à la section Référence des AWS SDK for Java 2.x API.

CreateLoadBalancer

L'exemple de code suivant montre comment utiliser `CreateLoadBalancer`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
```

```
        .subnets(subnetIdStrings)
        .name(lbName)
        .scheme("internet-facing")
        .build();

    // Create and wait for the load balancer to become available.
    CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
    String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

    ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
    DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

    System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
    WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Load Balancer " + lbName + " is available.");

    // Get the DNS name (endpoint) of the load balancer.
    String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
    System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

    // Create a listener for the load balance.
    Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

    CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();
```

```
        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateLoadBalancer](#) à la section Référence des AWS SDK for Java 2.x API.

CreateTargetGroup

L'exemple de code suivant montre comment utiliser `CreateTargetGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
```

```
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTargetGroup](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteLoadBalancer

L'exemple de code suivant montre comment utiliser `DeleteLoadBalancer`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
```

```
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteLoadBalancer](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteTargetGroup

L'exemple de code suivant montre comment utiliser `DeleteTargetGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
```

```
        .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTargetGroup](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeTargetHealth

L'exemple de code suivant montre comment utiliser `DescribeTargetHealth`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();
```



```
DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
return healthResponse.targetHealthDescriptions();
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeTargetHealth](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Créer et gérer un service résilient

L'exemple de code suivant montre comment créer un service Web à charge équilibrée qui renvoie des recommandations de livres, de films et de chansons. L'exemple montre comment le service répond aux défaillances et comment le restructurer pour accroître la résilience en cas de défaillance.

- Utilisez un groupe Amazon EC2 Auto Scaling pour créer des instances Amazon Elastic Compute Cloud (Amazon EC2) sur la base d'un modèle de lancement et pour maintenir le nombre d'instances dans une plage spécifiée.
- Gérez et distribuez les requêtes HTTP avec Elastic Load Balancing.
- Surveillez l'état des instances d'un groupe Auto Scaling et transférez les demandes uniquement aux instances saines.
- Exécutez un serveur Web Python sur chaque instance EC2 pour gérer les requêtes HTTP. Le serveur Web répond par des recommandations et des surveillances de l'état.
- Simulez un service de recommandation avec une table Amazon DynamoDB.
- Contrôlez la réponse du serveur Web aux demandes et aux contrôles de santé en mettant à jour AWS Systems Manager les paramètres.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    }
}
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("A - SETUP THE RESOURCES");
System.out.println("Press Enter when you're ready to start deploying
resources.");
in.nextLine();
deploy(loadBalancer);
System.out.println(DASHES);
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.
    To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
        """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
```

```

        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
        throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);
    }

```

```

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
        tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
            Creating an EC2 launch template that runs '{startup_script}' when an
            instance starts.
            This script starts a Python web server defined in the `server.py`
            script. The web server
            listens to HTTP requests on port 80 and responds to requests to '/'
            and to '/healthcheck'.
            For demo purposes, this server is run as the root user. In
            production, the best practice is to
            run a web server, such as Apache, with least-privileged credentials.

            The template also defines an IAM policy that each instance uses to
            assume a role that grants
            permissions to access the DynamoDB recommendation table and Systems
            Manager parameters
            that control the flow of the demo.
            """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
        startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "Creating an EC2 Auto Scaling group that maintains three EC2
            instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
        autoScalingGroupName);

        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
            starts, it listens for

```

```

        HTTP requests. You can see these instances in the console or
        continue with the demo.
        Press Enter when you're ready to continue.
        """);

    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating variables that control the flow of the demo.");
    ParameterHelper paramHelper = new ParameterHelper();
    paramHelper.reset();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an Elastic Load Balancing target group and load balancer.
    The target group
        defines how the load balancer connects to instances. The load
    balancer provides a
        single endpoint where clients connect and dispatches requests to
    instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {

```

```

        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {

```

```

        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        """);
}

```



```
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
    paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

    System.out.println(
        ""
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
    paramHelper.put(paramHelper.failureResponse, "static");

    System.out.println("""
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
    demoChoices(loadBalancer);

    System.out.println("Let's reinstate the recommendation service.");
    paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

    System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    AutoScaler autoScaler = new AutoScaler();
```

```
// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
+ " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");
```

```
        System.out.println("""
            Now, checking target health indicates that the instance with bad
credentials
            is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
            instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
            the load balancer takes unhealthy instances out of its rotation.
            """);

        demoChoices(loadBalancer);

        System.out.println(
            """
                Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
                """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
```

```
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
```

```

        new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
        Note that it can take a minute or two for the health
check to update
        after changes are made.
        """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {

```

```
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Créez une classe qui englobe les actions Auto Scaling et Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
```

```
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
    .build();

// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    }
}
```



```

    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.getInstanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
    .instanceIds(instanceId)
    .documentName("AWS-RunShellScript")
    .parameters(Collections.singletonMap("commands",
        Collections.singletonList("cd / && sudo python3 server.py
80")))
    .build();

getSSMClient().sendCommand(sendCommandRequest);
System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress) {
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,

```

```
    * and deletes all the resources.
    */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                .builder()
                .instanceProfileName(profileName)
                .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .roleName(roleName)
                .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(attachedPolicy.policyArn())
```

```

        .build();

        getIAMClient().detachRolePolicy(request);
        System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
    }

    getIAMClient().deleteRole(deleteRoleRequest);
    System.out.println("Instance profile and role deleted.");

} catch (IamException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 */

```

```
*
*/
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                }
            }
        }
    }
}
```

```
        portIsOpen = true;
    }

    if (!portIsOpen) {
        System.out
            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
    } else {
        break;
    }
}
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);
    }
}
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

    String availabilityZones = String.join(",", availabilityZoneNames);
    LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

    String[] zones = availabilityZones.split(",");
    CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);
    }
```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();
```

```
DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
    .filters(vpcFilter, azFilter, defaultForAZ)
    .build();

DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
subnets = response.subnets();
return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();
```



```

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
            .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                .builder()
                .policyArn(policy.arn())
                .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                    .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                    || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .roleName(roleName) // Specify the name of the IAM role
                        .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();

```

```
        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
System.out.println(InstanceProfile + " Deleted");
System.out.println("All roles and policies are deleted.");
}
}
```

Créez une classe qui englobe les actions Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
        DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
        getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
        DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
        getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```

```

    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {

```

```
boolean success = false;
int retries = 3;
CloseableHttpClient httpClient = HttpClients.createDefault();

// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
```

```

        .name(targetGroupName)
        .protocol(protocol)
        .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

```

```
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Créez une classe qui utilise DynamoDB pour simuler un service de recommandation.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /**
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```



```
* MediaType,
* forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();
    }
}
```

```
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
}
```

```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Créez une classe qui englobe les actions de Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)

- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

MediaStore exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with MediaStore.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateContainer

L'exemple de code suivant montre comment utiliser `CreateContainer`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
```

```
public static long sleepTime = 10;

public static void main(String[] args) {
    final String usage = ""

        Usage:    <containerName>

        Where:
            containerName - The name of the container to create.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String containerName = args[0];
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    createMediaContainer(mediaStoreClient, containerName);
    mediaStoreClient.close();
}

public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
    try {
        CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
            .containerName(containerName)
            .build();

        CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
        String status = containerResponse.container().status().toString();
        while (!status.equalsIgnoreCase("Active")) {
            status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
            System.out.println("Status - " + status);
            Thread.sleep(sleepTime * 1000);
        }
    }
}
```

```
        System.out.println("The container ARN value is " +
            containerResponse.container().arn());
        System.out.println("Finished ");

    } catch (MediaStoreException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateContainer](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteContainer

L'exemple de code suivant montre comment utiliser `DeleteContainer`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
        String containerName) {
        try {
            CreateContainerRequest containerRequest =
                CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
                mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
                    containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }
        }
    }
}
```



```
        System.out.println("The container ARN value is " +
containerResponse.container().arn());
        System.out.println("Finished ");

    } catch (MediaStoreException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteContainer](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteObject

L'exemple de code suivant montre comment utiliser `DeleteObject`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

            Usage:    <completePath> <containerName>

            Where:
                completePath - The path (including the container) of the item to
delete.
                containerName - The name of the container.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                .path(completePath)
                .build();

            mediaStoreData.deleteObject(deleteObjectRequest);
        }
    }
}
```

```
    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    mediaStoreClient.close();
    return response.container().endpoint();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeContainer

L'exemple de code suivant montre comment utiliser `DescribeContainer`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to describe.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
            containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
        containerName) {
```

```
    try {
        DescribeContainerRequest describeContainerRequest =
DescribeContainerRequest.builder()
            .containerName(containerName)
            .build();

        DescribeContainerResponse containerResponse =
mediaStoreClient.describeContainer(describeContainerRequest);
        System.out.println("The container name is " +
containerResponse.container().name());
        System.out.println("The container ARN is " +
containerResponse.container().arn());
        return containerResponse.container().status().toString();

    } catch (MediaStoreException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeContainer](#) à la section Référence des AWS SDK for Java 2.x API.

GetObject

L'exemple de code suivant montre comment utiliser `GetObject`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
```

```
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

            Usage:    <completePath> <containerName> <savePath>

            Where:
                completePath - The path of the object in the container (for
example, Videos5/sampleVideo.mp4).
                containerName - The name of the container.
                savePath - The path on the local drive where the file is saved,
including the file name (for example, C:/AWS/myvid.mp4).
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];
```

```
    Region region = Region.US_EAST_1;
    URI uri = new URI(getEndpoint(containerName));
    MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
        .endpointOverride(uri)
        .region(region)
        .build();

    getMediaObject(mediaStoreData, completePath, savePath);
    mediaStoreData.close();
}

public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

    try {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .path(completePath)
            .build();

        // Write out the data to a file.
        ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
        byte[] buffer = new byte[data.available()];
        data.read(buffer);

        File targetFile = new File(savePath);
        OutputStream outputStream = new FileOutputStream(targetFile);
        outputStream.write(buffer);
        System.out.println("The data was written to " + savePath);

    } catch (MediaStoreDataException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
```

```
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK for Java 2.x API.

ListContainers

L'exemple de code suivant montre comment utiliser `ListContainers`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```



```
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListContainers](#) à la section Référence des AWS SDK for Java 2.x API.

PutObject

L'exemple de code suivant montre comment utiliserPutObject.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = ""

                To run this example, supply the name of a container, a file location
                to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }
    }
}
```

```
    }

    String containerName = args[0];
    String filePath = args[1];
    String completePath = args[2];

    Region region = Region.US_EAST_1;
    URI uri = new URI(getEndpoint(containerName));
    MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
        .endpointOverride(uri)
        .region(region)
        .build();

    putMediaObject(mediaStoreData, filePath, completePath);
    mediaStoreData.close();
}

public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
filePath, String completePath) {
    try {
        File myFile = new File(filePath);
        RequestBody requestBody = RequestBody.fromFile(myFile);

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .path(completePath)
            .contentType("video/mp4")
            .build();

        PutObjectResponse response = mediaStoreData.putObject(objectRequest,
requestBody);
        System.out.println("The saved object is " +
response.storageClass().toString());

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getEndpoint(String containerName) {

    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
```

```
        .build();

        DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

        DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK for Java 2.x API.

OpenSearch Exemples de services utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du OpenSearch service AWS SDK for Java 2.x with.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateDomain

L'exemple de code suivant montre comment utiliser `CreateDomain`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to create.
        """;
    }
}
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        createNewDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void createNewDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                .dedicatedMasterEnabled(true)
                .dedicatedMasterCount(3)
                .dedicatedMasterType("t2.small.search")
                .instanceType("t2.small.search")
                .instanceCount(5)
                .build();

            EBSOptions ebsOptions = EBSOptions.builder()
                .ebsEnabled(true)
                .volumeSize(10)
                .volumeType(VolumeType.GP2)
                .build();

            NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
                .enabled(true)
                .build();

            CreateDomainRequest domainRequest = CreateDomainRequest.builder()
                .domainName(domainName)
                .engineVersion("OpenSearch_1.0")
                .clusterConfig(clusterConfig)
                .ebsOptions(ebsOptions)
                .nodeToNodeEncryptionOptions(encryptionOptions)
                .build();
```

```
        System.out.println("Sending domain creation request...");
        CreateDomainResponse createResponse =
searchClient.createDomain(domainRequest);
        System.out.println("Domain status is " +
createResponse.domainStatus().toString());
        System.out.println("Domain Id is " +
createResponse.domainStatus().domainId());

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDomain](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteDomain

L'exemple de code suivant montre comment utiliser `DeleteDomain`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteDomain {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <domainName>

            Where:
                domainName - The name of the domain to delete.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        deleteSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void deleteSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
                .domainName(domainName)
                .build();

            searchClient.deleteDomain(domainRequest);
            System.out.println(domainName + " was successfully deleted.");

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```



```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDomain](#) à la section Référence des AWS SDK for Java 2.x API.

ListDomainNames

L'exemple de code suivant montre comment utiliser `ListDomainNames`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.opensearch.OpenSearchClient;  
import software.amazon.awssdk.services.opensearch.model.DomainInfo;  
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;  
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesResponse;  
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;  
  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ListDomainNames {  
    public static void main(String[] args) {  
        Region region = Region.US_EAST_1;  
        OpenSearchClient searchClient = OpenSearchClient.builder()
```

```
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();
    listAllDomains(searchClient);
    System.out.println("Done");
}

public static void listAllDomains(OpenSearchClient searchClient) {
    try {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        ListDomainNamesResponse response =
searchClient.listDomainNames(namesRequest);
        List<DomainInfo> domainInfoList = response.domainNames();
        for (DomainInfo domain : domainInfoList)
            System.out.println("Domain name is " + domain.domainName());

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDomainNames](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateDomainConfig

L'exemple de code suivant montre comment utiliser `UpdateDomainConfig`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to update.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();

        updateSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }
}
```

```
public static void updateSpecificDomain(OpenSearchClient searchClient, String
domainName) {
    try {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        System.out.println("Sending domain update request...");
        UpdateDomainConfigResponse updateResponse =
searchClient.updateDomainConfig(updateDomainConfigRequest);
        System.out.println("Domain update response from Amazon OpenSearch
Service:");
        System.out.println(updateResponse.toString());

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateDomainConfig](#) à la section Référence des AWS SDK for Java 2.x API.

EventBridge exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with EventBridge.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.


Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour EventBridge

Les exemples de code suivants montrent comment commencer à utiliser EventBridge.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
            .region(region)
            .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
```

```
ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
    .limit(10)
    .build();

ListEventBusesResponse response =
eventBrClient.listEventBuses(busesRequest);
List<EventBus> buses = response.eventBuses();
for (EventBus bus : buses) {
    System.out.println("The name of the event bus is: " + bus.name());
    System.out.println("The ARN of the event bus is: " + bus.arn());
}

} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListEventBuses](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

DeleteRule

L'exemple de code suivant montre comment utiliser DeleteRule.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteRule](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeRule

L'exemple de code suivant montre comment utiliser `DescribeRule`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeRule](#) à la section Référence des AWS SDK for Java 2.x API.

DisableRule

L'exemple de code suivant montre comment utiliser `DisableRule`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Désactivez une règle à l'aide de son nom.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: " + eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();

            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: " + eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [DisableRule](#) à la section Référence des AWS SDK for Java 2.x API.

EnableRule

L'exemple de code suivant montre comment utiliser `EnableRule`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Activez une règle à l'aide de son nom.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String  
eventRuleName, Boolean isEnabled) {  
    try {  
        if (!isEnabled) {  
            System.out.println("Disabling the rule: " + eventRuleName);  
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()  
                .name(eventRuleName)  
                .build();  
  
            eventBrClient.disableRule(ruleRequest);  
        } else {  
            System.out.println("Enabling the rule: " + eventRuleName);  
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()  
                .name(eventRuleName)  
                .build();  
            eventBrClient.enableRule(ruleRequest);  
        }  
    }  
  
    } catch (EventBridgeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
    }  
}
```

```
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [EnableRule](#) à la section Référence des AWS SDK for Java 2.x API.

ListRuleNamesByTarget

L'exemple de code suivant montre comment utiliser `ListRuleNamesByTarget`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertoriez tous les noms de règle à l'aide de la cible.

```
public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
        .targetArn(topicArn)
        .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListRuleNamesByTarget](#) à la section Référence des AWS SDK for Java 2.x API.

ListRules

L'exemple de code suivant montre comment utiliser `ListRules`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Activez une règle à l'aide de son nom.

```
public static void listRules(EventBridgeClient eventBrClient) {
    try {
        ListRulesRequest rulesRequest = ListRulesRequest.builder()
            .eventBusName("default")
            .limit(10)
            .build();

        ListRulesResponse response = eventBrClient.listRules(rulesRequest);
        List<Rule> rules = response.rules();
        for (Rule rule : rules) {
            System.out.println("The rule name is : " + rule.name());
            System.out.println("The rule description is : " +
rule.description());
            System.out.println("The rule state is : " + rule.stateAsString());
        }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListRules](#) à la section Référence des AWS SDK for Java 2.x API.

ListTargetsByRule

L'exemple de code suivant montre comment utiliser `ListTargetsByRule`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertoriez toutes les cibles d'une règle à l'aide de son nom.

```
public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
    ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
        .rule(ruleName)
        .build();


    ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
    List<Target> targetsList = res.targets();
    for (Target target: targetsList) {
        System.out.println("Target ARN: "+target.arn());
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTargetsByRule](#) à la section Référence des AWS SDK for Java 2.x API.

PutEvents

L'exemple de code suivant montre comment utiliser `PutEvents`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \"" + email + "\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\" " +
        "}";

    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
        .source("ExampleSource")
        .detail(json)
        .detailType("ExampleType")
        .build();

    PutEventsRequest eventsRequest = PutEventsRequest.builder()
        .entries(entry)
        .build();

    eventBrClient.putEvents(eventsRequest);
}
```

- Pour plus de détails sur l'API, reportez-vous [PutEvents](#) à la section Référence des AWS SDK for Java 2.x API.

PutRule

L'exemple de code suivant montre comment utiliser `PutRule`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez une règle planifiée.

```
public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .scheduleExpression(cronExpression)
            .state("ENABLED")
            .description("A test rule that runs on a schedule created by the
Java API")
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Créez une règle qui se déclenche lorsqu'un objet est ajouté à un compartiment Amazon Simple Storage Service.

```
// Create a new event rule that triggers when an Amazon S3 object is created in
// a bucket.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
```

```
        "  \"source\": [\"aws.s3\"],\n" +\n"  \"detail-type\": [\"Object Created\"],\n" +\n"  \"detail\": {\n" +\n"    \"bucket\": {\n" +\n"      \"name\": [\"\" + bucketName + "\"]\n" +\n"    }\n" +\n"  }\n" +\n"}";\n\ntry {\n    PutRuleRequest ruleRequest = PutRuleRequest.builder()\n        .description("Created by using the AWS SDK for Java v2")\n        .name(eventRuleName)\n        .eventPattern(pattern)\n        .roleArn(roleArn)\n        .build();\n\n    PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);\n    System.out.println("The ARN of the new rule is " +\nruleResponse.ruleArn());\n\n    } catch (EventBridgeException e) {\n        System.err.println(e.awsErrorDetails().errorMessage());\n        System.exit(1);\n    }\n}
```

- Pour plus de détails sur l'API, reportez-vous [PutRule](#) à la section Référence des AWS SDK for Java 2.x API.

PutTargets

L'exemple de code suivant montre comment utiliser `PutTargets`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Ajoutez une rubrique Amazon SNS en tant que cible pour une règle.

```
// Add a rule which triggers an SNS target when a file is uploaded to an S3
// bucket.
public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
    String topicName, String eventRuleName, String bucketName) {
    String targetID = java.util.UUID.randomUUID().toString();
    Target myTarget = Target.builder()
        .id(targetID)
        .arn(topicArn)
        .build();

    List<Target> targets = new ArrayList<>();
    targets.add(myTarget);
    PutTargetsRequest request = PutTargetsRequest.builder()
        .eventBusName(null)
        .targets(targets)
        .rule(ruleName)
        .build();

    eventBrClient.putTargets(request);
    System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
        + bucketName + ".");
}
```

Ajoutez un transformateur d'entrée à une cible pour une règle.

```
public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
    String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    InputTransformer inputTransformer = InputTransformer.builder()
        .inputTemplate("\Notification: sample event was received.\")
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();
}
```



```
try {
    PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
        .rule(ruleName)
        .targets(target)
        .eventBusName(null)
        .build();

    eventBrClient.putTargets(targetsRequest);
} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutTargets](#) à la section Référence des AWS SDK for Java 2.x API.

RemoveTargets

L'exemple de code suivant montre comment utiliser `RemoveTargets`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Retirez toutes les cibles d'une règle à l'aide de son nom.

```
public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();

    ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
}
```

```
List<Target> allTargets = response.targets();

// Get all targets and delete them.
for (Target myTarget : allTargets) {
    RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
    .rule(eventRuleName)
    .ids(myTarget.id())
    .build();

    eventBrClient.removeTargets(removeTargetsRequest);
    System.out.println("Successfully removed the target");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [RemoveTargets](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Démarrer avec les règles et les cibles

L'exemple de code suivant illustre comment :

- Créez une règle et ajoutez-y une cible.
- Activez et désactivez les règles.
- Répertoriez et mettez à jour les règles et les cibles.
- Envoyez des événements, puis nettoyez les ressources.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java code example performs the following tasks:
*
* This Java V2 example performs the following tasks with Amazon EventBridge:
*
* 1. Creates an AWS Identity and Access Management (IAM) role to use with
* Amazon EventBridge.
* 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
* enabled.
* 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
* 4. Lists rules on the event bus.
* 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
* lets the user subscribe to it.
* 6. Adds a target to the rule that sends an email to the specified topic.
* 7. Creates an EventBridge event that sends an email when an Amazon S3 object
* is created.
* 8. Lists Targets.
* 9. Lists the rules for the same target.
* 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
* 11. Disables a specific rule.
* 12. Checks and print the state of the rule.
* 13. Adds a transform to the rule to change the text of the email.
* 14. Enables a specific rule.
* 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
* 16. Updates the rule to be a custom rule pattern.
* 17. Sending an event to trigger the rule.
* 18. Cleans up resources.
*
*/
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
    {
        final String usage = ""

                Usage:
                <roleName> <bucketName> <topicName> <eventRuleName>
```

```

        Where:
            roleName - The name of the role to create.
            bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name to create.
            topicName - The name of the Amazon Simple Notification Service
(Amazon SNS) topic to create.
            eventRuleName - The Amazon EventBridge rule name to create.
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String polJSON = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]}" +
        "};

    Scanner sc = new Scanner(System.in);
    String roleName = args[0];
    String bucketName = args[1];
    String topicName = args[2];
    String eventRuleName = args[3];

    Region region = Region.US_EAST_1;
    EventBridgeClient eventBrClient = EventBridgeClient.builder()
        .region(region)
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    Region regionGl = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(regionGl)

```

```
        .build();

    SnsClient snsClient = SnsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EventBridge example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out
        .println("1. Create an AWS Identity and Access Management (IAM) role
to use with Amazon EventBridge.");
    String roleArn = createIAMRole(iam, roleName, polJSON);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
    if (checkBucket(s3Client, bucketName)) {
        System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
        System.exit(1);
    }

    createBucket(s3Client, bucketName);
    Thread.sleep(3000);
    setBucketNotification(s3Client, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
    Thread.sleep(10000);
    addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. List rules on the event bus.");
    listRules(eventBrClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```
        System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
            "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email when
an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
System.out.println("11. Disable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, false);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Check and print the state of the rule.");
checkRule(eventBrClient, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Add a transform to the rule to change the text of
the email.");
updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Enable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, true);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 15. Trigger the updated rule by uploading a file to the
S3 bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 16. Update the rule to be a custom rule pattern.");
updateToCustomRule(eventBrClient, eventRuleName);
System.out.println("Updated event rule " + eventRuleName + " to use a custom
pattern.");
updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
System.out.println("Updated event target " + topicArn + ".");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
triggerCustomRule(eventBrClient, email);
System.out.println("Events have been sent. Press Enter to continue.");
sc.nextLine();
System.out.println(DASHES);
```

```

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
        IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
            .policyArn(policyArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()

```



```
        .roleName(roleName)
        .build();

    iam.deleteRole(roleRequest);
    System.out.println("*** Successfully deleted " + roleName);
}

public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
    // Remove all the objects from the S3 bucket.
    ListObjectsRequest listObjects = ListObjectsRequest.builder()
        .bucket(bucketName)
        .build();

    ListObjectsResponse res = s3Client.listObjects(listObjects);
    List<S3Object> objects = res.contents();
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

    for (S3Object myValue : objects) {
        toDelete.add(ObjectIdentifier.builder()
            .key(myValue.key())
            .build());
    }

    DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(Delete.builder()
            .objects(toDelete).build())
        .build();

    s3Client.deleteObjects(dor);

    // Delete the S3 bucket.
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();

    s3Client.deleteBucket(deleteBucketRequest);
    System.out.println("You have deleted the bucket and the objects");
}

// Delete the SNS topic.
public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
```

```
        .topicArn(topicArn)
        .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}

public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();

    ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
    List<Target> allTargets = response.targets();

    // Get all targets and delete them.
    for (Target myTarget : allTargets) {
        RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
            .rule(eventRuleName)
            .ids(myTarget.id())
            .build();

        eventBrClient.removeTargets(removeTargetsRequest);
        System.out.println("Successfully removed the target");
    }
}
```

```
    }  
  }  
  
  public static void triggerCustomRule(EventBridgeClient eventBrClient, String  
email) {  
    String json = "{" +  
      "\"UserEmail\": \"" + email + "\", " +  
      "\"Message\": \"This event was generated by example code.\", " +  
      "\"UtcTime\": \"Now.\""+  
      "}";  
  
    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()  
      .source("ExampleSource")  
      .detail(json)  
      .detailType("ExampleType")  
      .build();  
  
    PutEventsRequest eventsRequest = PutEventsRequest.builder()  
      .entries(entry)  
      .build();  
  
    eventBrClient.putEvents(eventsRequest);  
  }  
  
  public static void updateCustomRuleTargetWithTransform(EventBridgeClient  
eventBrClient, String topicArn,  
    String ruleName) {  
    String targetId = java.util.UUID.randomUUID().toString();  
    InputTransformer inputTransformer = InputTransformer.builder()  
      .inputTemplate("\"Notification: sample event was received.\"")  
      .build();  
  
    Target target = Target.builder()  
      .id(targetId)  
      .arn(topicArn)  
      .inputTransformer(inputTransformer)  
      .build();  
  
    try {  
      PutTargetsRequest targetsRequest = PutTargetsRequest.builder()  
        .rule(ruleName)  
        .targets(target)  
        .eventBusName(null)  
        .build();  
    }  
  }  
}
```

```
        eventBrClient.putTargets(targetsRequest);
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
    String customEventsPattern = "{" +
        "\"source\": [\"ExampleSource\"]," +
        "\"detail-type\": [\"ExampleType\"]" +
        "}";

    PutRuleRequest request = PutRuleRequest.builder()
        .name(ruleName)
        .description("Custom test rule")
        .eventPattern(customEventsPattern)
        .build();

    eventBrClient.putRule(request);
}

// Update an Amazon S3 object created rule with a transform on the target.
public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    Map<String, String> myMap = new HashMap<>();
    myMap.put("bucket", "$.detail.bucket.name");
    myMap.put("time", "$.time");

    InputTransformer inputTransformer = InputTransformer.builder()
        .inputTemplate("\"Notification: an object was uploaded to bucket
<bucket> at <time>.\")")
        .inputPathsMap(myMap)
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();
```

```
    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: " + eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();

            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: " + eventRuleName);
        }
    }
}
```

```
        EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
            .name(eventRuleName)
            .build();
        eventBrClient.enableRule(ruleRequest);
    }

} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

// Create and upload a file to an S3 bucket to trigger an event.
public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
    // Create a unique file name.
    String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
    String fileName = "TextFile" + fileSuffix + ".txt";

    File myFile = new File(fileName);
    FileWriter fw = new FileWriter(myFile.getAbsolutePath());
    BufferedWriter bw = new BufferedWriter(fw);
    bw.write("This is a sample file for testing uploads.");
    bw.close();

    try {
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        s3Client.putObject(putOb, RequestBody.fromFile(myFile));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
```

```

        .targetArn(topicArn)
        .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}

public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
    ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
        .rule(ruleName)
        .build();

    ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
    List<Target> targetsList = res.targets();
    for (Target target: targetsList) {
        System.out.println("Target ARN: "+target.arn());
    }
}

// Add a rule which triggers an SNS target when a file is uploaded to an S3
// bucket.
public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
    String topicName, String eventRuleName, String bucketName) {
    String targetID = java.util.UUID.randomUUID().toString();
    Target myTarget = Target.builder()
        .id(targetID)
        .arn(topicArn)
        .build();

    List<Target> targets = new ArrayList<>();
    targets.add(myTarget);
    PutTargetsRequest request = PutTargetsRequest.builder()
        .eventBusName(null)
        .targets(targets)
        .rule(ruleName)
        .build();

```

```
        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
        + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
    {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("email")
                .endpoint(email)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                .eventBusName("default")
                .limit(10)
                .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }

        } catch (EventBridgeException e) {
```



```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createSnsTopic(SnsClient snsClient, String topicName) {
    String topicPolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Sid\": \"EventBridgePublishTopic\"," +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Resource\": \"*\"," +
        "\"Action\": \"sns:Publish\"" +
        "}]}" +
        "}";

    Map<String, String> topicAttributes = new HashMap<>();
    topicAttributes.put("Policy", topicPolicy);
    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    System.out.println("Added topic " + topicName + " for email
subscriptions.");
    return response.topicArn();
}

// Create a new event rule that triggers when an Amazon S3 object is created in
// a bucket.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"" + bucketName + "\"]\n" +
        "    }\n" +
    }

```

```
        " }\n" +
        "}";

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .description("Created by using the AWS SDK for Java v2")
            .name(eventRuleName)
            .eventPattern(pattern)
            .roleArn(roleArn)
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

// Determine if the S3 bucket exists.
public static Boolean checkBucket(S3Client s3Client, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.headBucket(headBucketRequest);
        return true;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Set the S3 bucket notification configuration.
public static void setBucketNotification(S3Client s3Client, String bucketName) {
    try {
        EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
            .build();
```

```
NotificationConfiguration configuration =
NotificationConfiguration.builder()
    .eventBridgeConfiguration(eventBridgeConfiguration)
    .build();

PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
    .builder()
    .bucket(bucketName)
    .notificationConfiguration(configuration)
    .skipDestinationValidation(true)
    .build();

s3Client.putBucketNotificationConfiguration(configurationRequest);
System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static String createIAMRole(IamClient iam, String rolename, String  
  polJSON) {  
    try {  
      CreateRoleRequest request = CreateRoleRequest.builder()  
        .roleName(rolename)  
        .assumeRolePolicyDocument(polJSON)  
        .description("Created using the AWS SDK for Java")  
        .build();  
  
      CreateRoleResponse response = iam.createRole(request);  
      AttachRolePolicyRequest rolePolicyRequest =  
      AttachRolePolicyRequest.builder()  
        .roleName(rolename)  
        .policyArn("arn:aws:iam::aws:policy/  
      AmazonEventBridgeFullAccess")  
        .build();  
  
      iam.attachRolePolicy(rolePolicyRequest);  
      return response.role().arn();  
  
    } catch (IamException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
    return "";  
  }  
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [DeleteRule](#)
 - [DescribeRule](#)
 - [DisableRule](#)
 - [EnableRule](#)
 - [ListRuleNamesByTarget](#)
 - [ListRules](#)

- [ListTargetsByRule](#)
- [PutEvents](#)
- [PutRule](#)
- [PutTargets](#)

Exemples de prévisions utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK for Java 2.x with Forecast.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateDataset

L'exemple de code suivant montre comment utiliser CreateDataset.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <name>\s

            Where:
            name - The name of the data set.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
        forecast.close();
    }
}
```

```
}

public static String createForecastDataSet(ForecastClient forecast, String name)
{
    try {
        Schema schema = Schema.builder()
            .attributes(getSchema())
            .build();

        CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
            .datasetName(name)
            .domain("CUSTOM")
            .datasetType("RELATED_TIME_SERIES")
            .dataFrequency("D")
            .schema(schema)
            .build();

        CreateDatasetResponse response = forecast.createDataset(datasetRequest);
        return response.datasetArn();

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

// Create a SchemaAttribute list required to create a data set.
private static List<SchemaAttribute> getSchema() {

    List<SchemaAttribute> schemaList = new ArrayList<>();
    SchemaAttribute att1 = SchemaAttribute.builder()
        .attributeName("item_id")
        .attributeType("string")
        .build();

    SchemaAttribute att2 = SchemaAttribute.builder()
        .attributeName("timestamp")
        .attributeType("timestamp")
        .build();

    SchemaAttribute att3 = SchemaAttribute.builder()
        .attributeName("target_value")
```

```
        .attributeType("float")
        .build();

    // Push the SchemaAttribute objects to the List.
    schemaList.add(att1);
    schemaList.add(att2);
    schemaList.add(att3);
    return schemaList;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDataset](#) à la section Référence des AWS SDK for Java 2.x API.

CreateForecast

L'exemple de code suivant montre comment utiliser `CreateForecast`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```



```
public class CreateForecast {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <predictorArn>\s

            Where:
                name - The name of the forecast.\s
                predictorArn - The arn of the predictor to use.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }

    public static String createNewForecast(ForecastClient forecast, String name,
        String predictorArn) {
        try {
            CreateForecastRequest forecastRequest = CreateForecastRequest.builder()
                .forecastName(name)
                .predictorArn(predictorArn)
                .build();

            CreateForecastResponse response =
forecast.createForecast(forecastRequest);
            return response.forecastArn();

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateForecast](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteDataset

L'exemple de code suivant montre comment utiliser `DeleteDataset`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```

        <datasetARN>\s

        Where:
            datasetARN - The ARN of the data set to delete.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String datasetARN = args[0];
    Region region = Region.US_WEST_2;
    ForecastClient forecast = ForecastClient.builder()
        .region(region)
        .build();

    deleteForecastDataSet(forecast, datasetARN);
    forecast.close();
}

public static void deleteForecastDataSet(ForecastClient forecast, String
myDataSetARN) {
    try {
        DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
            .datasetArn(myDataSetARN)
            .build();

        forecast.deleteDataset(deleteRequest);
        System.out.println("The Data Set was deleted");

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Pour plus de détails sur l'API, reportez-vous [DeleteDataset](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteForecast

L'exemple de code suivant montre comment utiliser `DeleteForecast`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <datasetARN>\s

            Where:
                datasetARN - The ARN of the data set to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
```

```
    Region region = Region.US_WEST_2;
    ForecastClient forecast = ForecastClient.builder()
        .region(region)
        .build();

    deleteForecastDataSet(forecast, datasetARN);
    forecast.close();
}

public static void deleteForecastDataSet(ForecastClient forecast, String
myDataSetARN) {
    try {
        DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
            .datasetArn(myDataSetARN)
            .build();

        forecast.deleteDataset(deleteRequest);
        System.out.println("The Data Set was deleted");

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteForecast](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeForecast

L'exemple de code suivant montre comment utiliser `DescribeForecast`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <forecastarn>\s

                Where:
                forecastarn - The arn of the forecast (for example,
                "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String forecastarn = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        describe(forecast, forecastarn);
        forecast.close();
    }

    public static void describe(ForecastClient forecast, String forecastarn) {
        try {
```

```
        DescribeForecastRequest request = DescribeForecastRequest.builder()
            .forecastArn(forecastarn)
            .build();

        DescribeForecastResponse response = forecast.describeForecast(request);
        System.out.println("The name of the forecast is " +
            response.forecastName());

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeForecast](#) à la section Référence des AWS SDK for Java 2.x API.

ListDatasetGroups

L'exemple de code suivant montre comment utiliser `ListDatasetGroups`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                .maxResults(10)
                .build();

            ListDatasetGroupsResponse response = forecast.listDatasetGroups(group);
            List<DatasetGroupSummary> groups = response.datasetGroups();
            for (DatasetGroupSummary myGroup : groups) {
                System.out.println("The Data Set name is " +
myGroup.datasetGroupName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDatasetGroups](#) à la section Référence des AWS SDK for Java 2.x API.

ListForecasts

L'exemple de code suivant montre comment utiliser `ListForecasts`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListForecasts {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        listAllForecasts(forecast);
        forecast.close();
    }

    public static void listAllForecasts(ForecastClient forecast) {
        try {
            ListForecastsRequest request = ListForecastsRequest.builder()
```

```
        .maxResults(10)
        .build();

    ListForecastsResponse response = forecast.listForecasts(request);
    List<ForecastSummary> forecasts = response.forecasts();
    for (ForecastSummary forecastSummary : forecasts) {
        System.out.println("The name of the forecast is " +
forecastSummary.forecastName());
    }

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListForecasts](#) à la section Référence des AWS SDK for Java 2.x API.

AWS Glue exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with AWS Glue.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour AWS Glue

Les exemples de code suivants montrent comment démarrer avec AWS Glue.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
            .maxResults(10)
            .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListJobs](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateCrawler

L'exemple de code suivant montre comment utiliser `CreateCrawler`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;
import software.amazon.awssdk.services.glue.model.CrawlerTargets;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.S3Target;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <IAM> <s3Path> <cron> <dbName> <crawlerName>
```

```

        Where:
            IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
            s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *)).
            dbName - The database name.\s
            crawlerName - The name of the crawler.\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String iam = args[0];
    String s3Path = args[1];
    String cron = args[2];
    String dbName = args[3];
    String crawlerName = args[4];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
    glueClient.close();
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        // Add the S3Target to a list.

```

```
List<S3Target> targetList = new ArrayList<>();
targetList.add(s3Target);

CrawlerTargets targets = CrawlerTargets.builder()
    .s3Targets(targetList)
    .build();

CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
    .databaseName(dbName)
    .name(crawlerName)
    .description("Created by the AWS Glue Java API")
    .targets(targets)
    .role(iam)
    .schedule(cron)
    .build();

glueClient.createCrawler(crawlerRequest);
System.out.println(crawlerName + " was successfully created");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCrawler](#) à la section Référence des AWS SDK for Java 2.x API.

GetCrawler

L'exemple de code suivant montre comment utiliser `GetCrawler`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        getSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }
}
```

```
    }

    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
    {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            Instant createDate = response.crawler().creationTime();

            // Convert the Instant to readable date
            DateTimeFormatter formatter =
            DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                .withLocale(Locale.US)
                .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the Crawler is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetCrawler](#) à la section Référence des AWS SDK for Java 2.x API.

GetDatabase

L'exemple de code suivant montre comment utiliser `GetDatabase`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetDatabase {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <databaseName>

                Where:
                databaseName - The name of the database.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String databaseName = args[0];
Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();

getSpecificDatabase(glueClient, databaseName);
glueClient.close();
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetDatabase](#) à la section Référence des AWS SDK for Java 2.x API.

GetTables

L'exemple de code suivant montre comment utiliser `GetTables`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbName> <tableName>

                Where:
                dbName - The database name.\s
                tableName - The name of the table.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbName = args[0];
        String tableName = args[1];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();

        getGlueTable(glueClient, dbName, tableName);
        glueClient.close();
    }

    public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName) {
        try {
            GetTableRequest tableRequest = GetTableRequest.builder()
                .databaseName(dbName)
                .name(tableName)
                .build();

            GetTableResponse tableResponse = glueClient.getTable(tableRequest);
            Instant createDate = tableResponse.table().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                .withLocale(Locale.US)
                .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the table is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetTables](#) à la section Référence des AWS SDK for Java 2.x API.

StartCrawler

L'exemple de code suivant montre comment utiliser `StartCrawler`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String crawlerName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    startSpecificCrawler(glueClient, crawlerName);
    glueClient.close();
}

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [StartCrawler](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Premiers pas avec les Crawlers et les tâches

L'exemple de code suivant illustre comment :

- Créez un Crawler qui indexe un compartiment Amazon S3 public et génère une base de données de métadonnées au format CSV.
- Répertoirez les informations relatives aux bases de données et aux tables de votre AWS Glue Data Catalog.

- Créez une tâche pour extraire les données CSV du compartiment S3, transformer les données et charger la sortie au format JSON dans un autre compartiment S3.
- Répertoriez les informations relatives aux exécutions de tâches, visualisez les données transformées et nettoyez les ressources.

Pour plus d'informations, consultez [Tutoriel : prise en main de AWS Glue Studio](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
```

```
* 12. Delete a database.
* 13. Delete a crawler.
*/
```

```
public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
                jobName - The name you assign to this job definition.
                scriptLocation - The Amazon S3 path to a script that runs a job.
                locationUri - The location of the database
                bucketNameSc - The Amazon S3 bucket name used when creating a
job

            """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];
```



```
Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();
System.out.println(DASHES);
System.out.println("Welcome to the AWS Glue scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a database.");
createDatabase(glueClient, dbName, locationUri);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a crawler.");
createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get a crawler.");
getSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Start a crawler.");
startSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a database.");
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait 5 min for the tables to become available");
TimeUnit.MINUTES.sleep(5);
System.out.println("6. Get tables.");
String myTableName = getGlueTables(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a job.");
createJob(glueClient, jobName, iam, scriptLocation);
System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        getAllJobs(glueClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        getJobRuns(glueClient, jobName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        deleteJob(glueClient, jobName);
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        deleteDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
```

```
        .locationUri(locationUri)
        .build();

    CreateDatabaseRequest request = CreateDatabaseRequest.builder()
        .databaseInput(input)
        .build();

    glueClient.createDatabase(request);
    System.out.println(dbName + " was successfully created");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);
        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
    }
}
```

```
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        boolean ready = false;
        while (!ready) {
            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            String status = response.crawler().stateAsString();
            if (status.compareTo("READY") == 0) {
                ready = true;
            }
            Thread.sleep(3000);
        }

        System.out.println("The crawler is now ready");

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully started!");

    } catch (GlueException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getGlueTables(GlueClient glueClient, String dbName) {
    String myTableName = "";
    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        if (tables.isEmpty()) {
            System.out.println("No tables were returned");
        } else {
            for (Table table : tables) {
                myTableName = table.name();
            }
        }
    }
}
```

```
        System.out.println("Table name is: " + myTableName);
    }
}

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return myTableName;
}

public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
    String outBucket) {
    try {
        Map<String, String> myMap = new HashMap<>();
        myMap.put("--input_database", inputDatabase);
        myMap.put("--input_table", inputTable);
        myMap.put("--output_bucket_url", outBucket);

        StartJobRunRequest runRequest = StartJobRunRequest.builder()
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .arguments(myMap)
            .jobName(jobName)
            .build();

        StartJobRunResponse response = glueClient.startJobRun(runRequest);
        System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
    try {
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
            .name("glueetl")
            .scriptLocation(scriptLocation)
```

```
        .build();

        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .description("A Job created by using the AWS SDK for Java V2")
            .glueVersion("2.0")
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .name(jobName)
            .role(iam)
            .command(command)
            .build();

        glueClient.createJob(jobRequest);
        System.out.println(jobName + " was successfully created.");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAllJobs(GlueClient glueClient) {
    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job : jobs) {
            System.out.println("Job name is : " + job.name());
            System.out.println("The job worker type is : " +
job.workerType().name());
        }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getJobRuns(GlueClient glueClient, String jobName) {
    try {
        GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
```

```
        .jobName(jobName)
        .maxResults(20)
        .build();

    boolean jobDone = false;
    while (!jobDone) {
        GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
        List<JobRun> jobRuns = response.jobRuns();
        for (JobRun jobRun : jobRuns) {
            String jobState = jobRun.jobRunState().name();
            if (jobState.compareTo("SUCCEEDED") == 0) {
                System.out.println(jobName + " has succeeded");
                jobDone = true;
            } else if (jobState.compareTo("STOPPED") == 0) {
                System.out.println("Job run has stopped");
                jobDone = true;
            } else if (jobState.compareTo("FAILED") == 0) {
                System.out.println("Job run has failed");
                jobDone = true;
            } else if (jobState.compareTo("TIMEOUT") == 0) {
                System.out.println("Job run has timed out");
                jobDone = true;
            } else {
                System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                System.out.println("Job run Id is " + jobRun.id());
                System.out.println("The Glue version is " +
jobRun.glueVersion());
            }
            TimeUnit.SECONDS.sleep(5);
        }
    }

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteJob(GlueClient glueClient, String jobName) {
```



```
    try {
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDatabase(GlueClient glueClient, String databaseName) {
    try {
        DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
            .name(databaseName)
            .build();

        glueClient.deleteDatabase(request);
        System.out.println(databaseName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.deleteCrawler(deleteCrawlerRequest);
        System.out.println(crawlerName + " was deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)
 - [GetJob](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)
 - [StartCrawler](#)
 - [StartJobRun](#)

HealthImaging exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with HealthImaging.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CopyImageSet

L'exemple de code suivant montre comment utiliser `CopyImageSet`.

SDK pour Java 2.x

```
public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imageSetId,
    String latestVersionId,
    String destinationImageSetId,
    String destinationVersionId) {

    try {
        CopySourceImageSetInformation copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
            .latestVersionId(latestVersionId)
            .build();

        CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
            .sourceImageSet(copySourceImageSetInformation);

        if (destinationImageSetId != null) {
            copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                .imageSetId(destinationImageSetId)
                .latestVersionId(destinationVersionId)
                .build());
        }

        CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
```

```
        .datastoreId(datastoreId)
        .sourceImageSetId(imageSetId)
        .copyImageSetInformation(copyImageSetBuilder.build())
        .build();

    CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

    return response.destinationImageSetProperties().imageSetId();
} catch (MedicalImagingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyImageSet](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

CreateDatastore

L'exemple de code suivant montre comment utiliser `CreateDatastore`.

SDK pour Java 2.x

```
public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
    String datastoreName) {
    try {
        CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
            .datastoreName(datastoreName)
            .build();
```

```
        CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
        return response.datastoreId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDatastore](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


DeleteDatastore

L'exemple de code suivant montre comment utiliser `DeleteDatastore`.

SDK pour Java 2.x

```
public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
        String datastoreID) {
    try {
        DeleteDatastoreRequest datastoreRequest =
DeleteDatastoreRequest.builder()
            .datastoreId(datastoreID)
            .build();
        medicalImagingClient.deleteDatastore(datastoreRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDatastore](#) à la section Référence des AWS SDK for Java 2.x API.

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

DeleteImageSet

L'exemple de code suivant montre comment utiliser DeleteImageSet.

SDK pour Java 2.x

```
public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        medicalImagingClient.deleteImageSet(deleteImageSetRequest);

        System.out.println("The image set was deleted.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteImageSet](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

GetDICOMImportJob

L'exemple de code suivant montre comment utiliser `GetDICOMImportJob`.

SDK pour Java 2.x

```
public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
                String datastoreId,
                String jobId) {

    try {
        GetDicomImportJobRequest getDicomImportJobRequest =
        GetDicomImportJobRequest.builder()
            .datastoreId(datastoreId)
            .jobId(jobId)
            .build();

        GetDicomImportJobResponse response =
        medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
        return response.jobProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Pour plus de détails sur l'API, consultez [GetDICOM ImportJob](#) dans la section AWS SDK for Java 2.x API Reference.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

GetDatastore

L'exemple de code suivant montre comment utiliser `GetDatastore`.

SDK pour Java 2.x

```
public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
    String datastoreID) {
    try {
        GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
            .datastoreId(datastoreID)
            .build();
        GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
        return response.datastoreProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetDatastore](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

GetImageFrame

L'exemple de code suivant montre comment utiliser `GetImageFrame`.

SDK pour Java 2.x

```
public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
        String destinationPath,
        String datastoreId,
        String imagesetId,
        String imageFrameId) {

    try {
        GetImageFrameRequest getImageSetMetadataRequest =
        GetImageFrameRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .imageFrameInformation(ImageFrameInformation.builder()
                .imageFrameId(imageFrameId)
                .build())
            .build();

        medicalImagingClient.getImageFrame(getImageSetMetadataRequest,
        FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Image frame downloaded to " +
        destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetImageFrame](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

GetImageSet

L'exemple de code suivant montre comment utiliser `GetImageSet`.

SDK pour Java 2.x

```
public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imagesetId,
    String versionId) {
    try {
        GetImageSetRequest.Builder getImageSetRequestBuilder =
        GetImageSetRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetRequestBuilder =
            getImageSetRequestBuilder.versionId(versionId);
        }

        return
        medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetImageSet](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

GetImageSetMetadata

L'exemple de code suivant montre comment utiliser `GetImageSetMetadata`.

SDK pour Java 2.x

```
public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
    String destinationPath,
    String datastoreId,
    String imagesetId,
    String versionId) {


    try {
        GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
        GetImageSetMetadataRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetMetadataRequestBuilder =
            getImageSetMetadataRequestBuilder.versionId(versionId);
        }

        medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
            FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Metadata downloaded to " + destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetImageSetMetadata](#) à la section Référence des AWS SDK for Java 2.x API.

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

ListDICOMImportJobs

L'exemple de code suivant montre comment utiliser `ListDICOMImportJobs`.

SDK pour Java 2.x

```
public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
                    String datastoreId) {

    try {
        ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                            .datastoreId(datastoreId)
                            .build();
        ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
        return response.jobSummaries();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return new ArrayList<>();
}
```

- Pour plus de détails sur l'API, voir [ListDicom ImportJobs dans la référence](#) des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

ListDatastores

L'exemple de code suivant montre comment utiliser `ListDatastores`.

SDK pour Java 2.x

```
public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
    try {
        ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
            .build();
        ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
        List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

        responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

        return datastoreSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDatastores](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

ListImageSetVersions

L'exemple de code suivant montre comment utiliser `ListImageSetVersions`.

SDK pour Java 2.x

```
public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        ListImageSetVersionsIterable responses = medicalImagingClient
            .listImageSetVersionsPaginator(getImageSetRequest);
        List<ImageSetProperties> imageSetProperties = new ArrayList<>();
        responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

        return imageSetProperties;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [ListImageSetVersions](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

ListTagsForResource

L'exemple de code suivant montre comment utiliser `ListTagsForResource`.

SDK pour Java 2.x

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTagsForResource](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

SearchImageSets

L'exemple de code suivant montre comment utiliser `SearchImageSets`.

SDK pour Java 2.x

Fonction utilitaire permettant de rechercher des ensembles d'images.

```

public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
    MedicalImagingClient medicalImagingClient,
    String datastoreId, SearchCriteria searchCriteria) {
    try {
        SearchImageSetsRequest dataStoreRequest =
SearchImageSetsRequest.builder()
            .datastoreId(datastoreId)
            .searchCriteria(searchCriteria)
            .build();
        SearchImageSetsIterable responses = medicalImagingClient
            .searchImageSetsPaginator(dataStoreRequest);
        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();

        responses.stream().forEach(response -> imageSetsMetadataSummaries
            .addAll(response.imageSetsMetadataSummaries()));

        return imageSetsMetadataSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

```

Cas d'utilisation #1 : opérateur EQUAL.

```

List<SearchFilter> searchFilters =
Collections.singletonList(SearchFilter.builder()
    .operator(Operator.EQUAL)
    .values(SearchByAttributeValue.builder()
        .dicomPatientId(patientId)
        .build())
    .build());

SearchCriteria searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .build();

List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
searchMedicalImagingImageSets(

```



```

        medicalImagingClient,
        datastoreId, searchCriteria);
    if (imageSetsMetadataSummaries != null) {
        System.out.println("The image sets for patient " + patientId + " are:\n"
            + imageSetsMetadataSummaries);
        System.out.println();
    }
}

```

Cas d'utilisation #2 : opérateur BETWEEN utilisant DICOM StudyDate et StudyTime DICOM.

```

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
searchFilters = Collections.singletonList(SearchFilter.builder()
    .operator(Operator.BETWEEN)
    .values(SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
    .dicomStudyDate("19990101")
    .dicomStudyTime("000000.000")
    .build())
    .build(),
    SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
    .dicomStudyDate((LocalDate.now()
        .format(formatter)))
    .dicomStudyTime("000000.000")
    .build())
    .build())
    .build());

searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .build();

imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
    datastoreId, searchCriteria);
    if (imageSetsMetadataSummaries != null) {
        System.out.println(
            "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
            +

```

```

        imageSetsMetadataSummaries);
    System.out.println();
}

```

Cas d'utilisation #3 : opérateur BETWEEN utilisant CreatedAt. Les études temporelles étaient auparavant poursuivies.

```

searchFilters = Collections.singletonList(SearchFilter.builder()
    .operator(Operator.BETWEEN)
    .values(SearchByAttributeValue.builder()
        .createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
        .build(),
        SearchByAttributeValue.builder()
        .createdAt(Instant.now())
        .build())
    .build());

searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .build();

imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
    datastoreId, searchCriteria);
if (imageSetsMetadataSummaries != null) {
    System.out.println("The image sets searched with BETWEEN operator using
createdAt are:\n "
        + imageSetsMetadataSummaries);
    System.out.println();
}

```

Cas d'utilisation #4 : opérateur EQUAL sur l'SeriesInstanceUID DICOM et BETWEEN sur UpdateDat et tri la réponse dans l'ordre ASC sur le champ UpdatedAt.

```

Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
Instant endDate = Instant.now();

searchFilters = Arrays.asList(
    SearchFilter.builder()
        .operator(Operator.EQUAL)
        .values(SearchByAttributeValue.builder()

```

```

        .dicomSeriesInstanceUID(seriesInstanceUID)
        .build())
        .build(),
    SearchFilter.builder()
        .operator(Operator.BETWEEN)
        .values(

SearchByAttributeValue.builder().updatedAt(startDate).build(),

SearchByAttributeValue.builder().updatedAt(endDate).build()
        ).build());

    Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

    searchCriteria = SearchCriteria.builder()
        .filters(searchFilters)
        .sort(sort)
        .build();

    imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
        datastoreId, searchCriteria);
    if (imageSetsMetadataSummaries != null) {
        System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
            "in ASC order on updatedAt field are:\n "
            + imageSetsMetadataSummaries);
        System.out.println();
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [SearchImageSets](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

StartDICOMImportJob

L'exemple de code suivant montre comment utiliser `StartDICOMImportJob`.

SDK pour Java 2.x

```
public static String startDicomImportJob(MedicalImagingClient
medicalImagingClient,
    String jobName,
    String datastoreId,
    String dataAccessRoleArn,
    String inputS3Uri,
    String outputS3Uri) {

    try {
        StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
            .jobName(jobName)
            .datastoreId(datastoreId)
            .dataAccessRoleArn(dataAccessRoleArn)
            .inputS3Uri(inputS3Uri)
            .outputS3Uri(outputS3Uri)
            .build();

        StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
        return response.jobId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- Pour plus de détails sur l'API, voir [StartDICOM ImportJob dans la référence](#) des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

TagResource

L'exemple de code suivant montre comment utiliser `TagResource`.

SDK pour Java 2.x

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [TagResource](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

UntagResource

L'exemple de code suivant montre comment utiliser `UntagResource`.

SDK pour Java 2.x

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
        String resourceArn,
        Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tagKeys(tagKeys)
            .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UntagResource](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

UpdateImageSetMetadata

L'exemple de code suivant montre comment utiliser UpdateImageSetMetadata.

SDK pour Java 2.x

```
public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
        String datastoreId,
        String imagesetId,
```

```

String versionId,
MetadataUpdates

metadataUpdates) {
    try {
        UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
            .builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .latestVersionId(versionId)
            .updateImageSetMetadataUpdates(metadataUpdates)
            .build();

        UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

        System.out.println("The image set metadata was updated" + response);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Cas d'utilisation #1 : Insérer ou mettre à jour un attribut.

```

final String insertAttributes = ""
    {
        "SchemaVersion": 1.1,
        "Study": {
            "DICOM": {
                "StudyDescription": "CT CHEST"
            }
        }
    }
    "";
MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
    .dicomUpdates(DICOMUpdates.builder()
        .updateableAttributes(SdkBytes.fromByteBuffer(
            ByteBuffer.wrap(insertAttributes
                .getBytes(StandardCharsets.UTF_8))))
        .build())
    .build();

```

```

        updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
            imagesetId,
                versionid, metadataInsertUpdates);

```

Cas d'utilisation #2 : Supprimer un attribut.

```

        final String removeAttributes = ""
            {
                "SchemaVersion": 1.1,
                "Study": {
                    "DICOM": {
                        "StudyDescription": "CT CHEST"
                    }
                }
            }
        """;
        MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
            .dicomUpdates(DICOMUpdates.builder()
                .removableAttributes(SdkBytes.fromByteBuffer(
                    ByteBuffer.wrap(removeAttributes
                        .getBytes(StandardCharsets.UTF_8))))
                .build())
            .build();

        updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
            imagesetId,
                versionid, metadataRemoveUpdates);

```

Cas d'utilisation #3 : Supprimer une instance.

```

        final String removeInstance = ""
            {
                "SchemaVersion": 1.1,
                "Study": {
                    "Series": {
                        "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                            "Instances": {
                                "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1":
                                    {}
                            }
                        }
                    }
                }
            }
        """;

```



```
        }
    }
}
}
}
}";
MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
    .dicomUpdates(DICOMUpdates.builder()
        .removableAttributes(SdkBytes.fromByteBuffer(
            ByteBuffer.wrap(removeInstance
                .getBytes(StandardCharsets.UTF_8))))
        .build())
    .build();

updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
    imagesetId,
    versionid, metadataRemoveUpdates);
```

- Pour plus de détails sur l'API, reportez-vous [UpdateImageSetMetadata](#) à la section Référence des AWS SDK for Java 2.x API.

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Scénarios

Marquage d'un magasin de données

L'exemple de code suivant montre comment étiqueter un magasin de HealthImaging données.

SDK pour Java 2.x

Pour étiqueter un magasin de données.

```
        final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

        TagResource.tagMedicalImagingResource(medicalImagingClient,
            datastoreArn,
                ImmutableMap.of("Deployment", "Development"));
```

Fonction utilitaire permettant de baliser une ressource.

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Pour répertorier les balises d'un magasin de données.

```
final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

ListTagsForResourceResponse result =
ListTagsForResource.listMedicalImagingResourceTags(
    medicalImagingClient,
    datastoreArn);
if (result != null) {
    System.out.println("Tags for resource: " + result.tags());
}
```

Fonction utilitaire permettant de répertorier les balises d'une ressource.

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
```

```
String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

Pour supprimer le balisage d'un magasin de données.

```
final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

UntagResource.untagMedicalImagingResource(medicalImagingClient,
datastoreArn,
    Collections.singletonList("Deployment"));
```


Fonction utilitaire permettant de détaguer une ressource.

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tagKeys(tagKeys)
            .build();

        medicalImagingClient.untagResource(untagResourceRequest);
    }
```

```
        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Marquer un ensemble d'images

L'exemple de code suivant montre comment baliser un ensemble HealthImaging d'images.

SDK pour Java 2.x

Pour baliser un ensemble d'images.

```
        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

        TagResource.tagMedicalImagingResource(medicalImagingClient,
imageSetArn,
                                           ImmutableMap.of("Deployment", "Development"));
```

Fonction utilitaire permettant de baliser une ressource.

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
```

```
        String resourceArn,
        Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Pour répertorier les balises d'un ensemble d'images.

```
        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

        ListTagsForResourceResponse result =
        ListTagsForResource.listMedicalImagingResourceTags(
            medicalImagingClient,
            imageSetArn);
        if (result != null) {
            System.out.println("Tags for resource: " + result.tags());
        }
}
```

Fonction utilitaire permettant de répertorier les balises d'une ressource.

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
        ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();
    }
}
```

```

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

```

Pour annuler le balisage d'un ensemble d'images.

```

        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

        UntagResource.untagMedicalImagingResource(medicalImagingClient,
imageSetArn,
            Collections.singletonList("Deployment"));

```

Fonction utilitaire permettant de détaguer une ressource.

```

public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tagKeys(tagKeys)
            .build();


        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

```
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exemples d'IAM utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK for Java 2.x with IAM.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour IAM

Les exemples de code suivants montrent comment démarrer avec IAM.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```


- Pour plus de détails sur l'API, reportez-vous [ListPolicies](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

AttachRolePolicy

L'exemple de code suivant montre comment utiliser `AttachRolePolicy`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AttachRolePolicy {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <roleName> <policyArn>\s

    Where:
        roleName - A role name that you can obtain from the AWS
Management Console.\s
        policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String roleName = args[0];
String policyArn = args[1];

Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

attachIAMRolePolicy(iam, roleName, policyArn);
iam.close();
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy : attachedPolicies) {
```

```
        polArn = policy.policyArn();
        if (polArn.compareTo(policyArn) == 0) {
            System.out.println(roleName + " policy is already attached to
this role.");
            return;
        }
    }

    AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

    iam.attachRolePolicy(attachRequest);

    System.out.println("Successfully attached policy " + policyArn +
        " to role " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Done");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [AttachRolePolicy](#) à la section Référence des AWS SDK for Java 2.x API.

CreateAccessKey

L'exemple de code suivant montre comment utiliser `CreateAccessKey`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <user>\s

                Where:
                user - An AWS IAM user that you can obtain from the AWS
Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
```

```
    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey().accessKeyId();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateAccessKey](#) à la section Référence des AWS SDK for Java 2.x API.

CreateAccountAlias

L'exemple de code suivant montre comment utiliser `CreateAccountAlias`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <alias>\s

            Where:
                alias - The account alias to create (for example, myawsaccount).
\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IAMClient iam = IAMClient.builder()
            .region(region)
            .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IAMClient iam, String alias) {
        try {
            CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
                .accountAlias(alias)
                .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

        } catch (IAMException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateAccountAlias](#) à la section Référence des AWS SDK for Java 2.x API.

CreatePolicy

L'exemple de code suivant montre comment utiliser `CreatePolicy`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;  
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;  
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;  
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;  
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;  
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.waiters.IamWaiter;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class CreatePolicy {  
  
    public static final String PolicyDocument = "{" +  
        "  \"Version\": \"2012-10-17\", " +
```

```

    "  \"Statement\": [" +
    "    {" +
    "      \"Effect\": \"Allow\", " +
    "      \"Action\": [" +
    "        \"dynamodb:DeleteItem\", " +
    "        \"dynamodb:GetItem\", " +
    "        \"dynamodb:PutItem\", " +
    "        \"dynamodb:Scan\", " +
    "        \"dynamodb:UpdateItem\"" +
    "      ], " +
    "      \"Resource\": \"*\":" +
    "    }" +
    "  ]" +
    "};

```

```

public static void main(String[] args) {

    final String usage = ""
        Usage:
            CreatePolicy <policyName>\s

        Where:
            policyName - A unique policy name.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String policyName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMPolicy(iam, policyName);
    System.out.println("Successfully created a policy with this ARN value: " +
result);
    iam.close();
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {

```



```
// Create an IamWaiter object.
IamWaiter iamWaiter = iam.waiter();

CreatePolicyRequest request = CreatePolicyRequest.builder()
    .policyName(policyName)
    .policyDocument(PolicyDocument)
    .build();

CreatePolicyResponse response = iam.createPolicy(request);

// Wait until the policy is created.
GetPolicyRequest polRequest = GetPolicyRequest.builder()
    .policyArn(response.policy().arn())
    .build();

WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
return response.policy().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreatePolicy](#) à la section Référence des AWS SDK for Java 2.x API.

CreateRole

L'exemple de code suivant montre comment utiliser `CreateRole`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
 * This example requires a trust policy document. For more information, see:
 * https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
 *
 * In addition, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = ""
            Usage:
                <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that represents
the trust policy.\s
            """;

        if (args.length != 2) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String rolename = args[0];
    String fileLocation = args[1];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMRole(iam, rolename, fileLocation);
    System.out.println("Successfully created user: " + result);
    iam.close();
}

public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " + response.role().arn());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateRole](#) à la section Référence des AWS SDK for Java 2.x API.

CreateUser

L'exemple de code suivant montre comment utiliser `CreateUser`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <username>\s

                Where:
                username - The name of the user to create.\s
    }
```

```
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMUser(iam, username);
    System.out.println("Successfully created user: " + result);
    iam.close();
}

public static String createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created.
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateUser](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteAccessKey

L'exemple de code suivant montre comment utiliser `DeleteAccessKey`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;  
import software.amazon.awssdk.services.iam.model.IamException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DeleteAccessKey {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <username> <accessKey>\s  
  
            Where:  
            username - The name of the user.\s
```

```
        accessKey - The access key ID for the secret access key you want
to delete.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    String accessKey = args[1];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();
    deleteKey(iam, username, accessKey);
    iam.close();
}

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAccessKey](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteAccountAlias

L'exemple de code suivant montre comment utiliser `DeleteAccountAlias`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <alias>\s

            Where:
                alias - The account alias to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
```



```
IamClient iam = IamClient.builder()
    .region(region)
    .build();

deleteIAMAccountAlias(iam, alias);
iam.close();
}

public static void deleteIAMAccountAlias(IamClient iam, String alias) {
    try {
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.deleteAccountAlias(request);
        System.out.println("Successfully deleted account alias " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAccountAlias](#) à la section Référence des AWS SDK for Java 2.x API.

DeletePolicy

L'exemple de code suivant montre comment utiliser DeletePolicy.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <policyARN>\s

            Where:
                policyARN - A policy ARN value to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(policyARN)

```

```
        .build();

        iam.deletePolicy(request);
        System.out.println("Successfully deleted the policy");

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeletePolicy](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteUser

L'exemple de code suivant montre comment utiliser `DeleteUser`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
*/
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <userName>\s

            Where:
                userName - The name of the user to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                .userName(userName)
                .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteUser](#) à la section Référence des AWS SDK for Java 2.x API.

DetachRolePolicy

L'exemple de code suivant montre comment utiliser `DetachRolePolicy`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <roleName> <policyArn>\s

                Where:
                roleName - A role name that you can obtain from the AWS
                Management Console.\s
    }
}
```

```
        policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String roleName = args[0];
    String policyArn = args[1];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();
    detachPolicy(iam, roleName, policyArn);
    System.out.println("Done");
    iam.close();
}

public static void detachPolicy(IamClient iam, String roleName, String
policyArn) {
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DetachRolePolicy](#) à la section Référence des AWS SDK for Java 2.x API.

ListAccessKeys

L'exemple de code suivant montre comment utiliser `ListAccessKeys`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <userName>\s

                Where:
                userName - The name of the user for which access keys are
retrieved.\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String userName = args[0];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

listKeys(iam, userName);
System.out.println("Done");
iam.close();
}

public static void listKeys(IamClient iam, String userName) {
    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if (newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
                    .build();

                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {

```



```
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListAccessKeys](#) à la section Référence des AWS SDK for Java 2.x API.

ListAccountAliases

L'exemple de code suivant montre comment utiliser `ListAccountAliases`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccountAliases {
```

```
public static void main(String[] args) {
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    listAliases(iam);
    System.out.println("Done");
    iam.close();
}

public static void listAliases(IamClient iam) {
    try {
        ListAccountAliasesResponse response = iam.listAccountAliases();
        for (String alias : response.accountAliases()) {
            System.out.printf("Retrieved account alias %s", alias);
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListAccountAliases](#) à la section Référence des AWS SDK for Java 2.x API.

ListUsers

L'exemple de code suivant montre comment utiliser `ListUsers`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request = ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                        .marker(newMarker)
                        .build();

                    response = iam.listUsers(request);
                }
            }
        }
    }
}
```

```
        }

        for (User user : response.users()) {
            System.out.format("\n Retrieved user %s", user.userName());
            AttachedPermissionsBoundary permissionsBoundary =
user.permissionsBoundary();
            if (permissionsBoundary != null)
                System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
        }

        if (!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListUsers](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateAccessKey

L'exemple de code suivant montre comment utiliser `UpdateAccessKey`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username> <accessId> <status>\s

            Where:
                username - The name of the user whose key you want to update.\s
                accessId - The access key ID of the secret access key you want
to update.\s
                status - The status you want to assign to the secret access key.
\s

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
```

```

        .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
        iam.close();
    }

    public static void updateKey(IamClient iam, String username, String accessId,
String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                .accessKeyId(accessId)
                .userName(username)
                .status(statusType)
                .build();

            iam.updateAccessKey(request);
            System.out.printf("Successfully updated the status of access key %s to"
+
                "status %s for user %s", accessId, status, username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [UpdateAccessKey](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateUser

L'exemple de code suivant montre comment utiliser `UpdateUser`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <curName> <newName>\s

                Where:
                curName - The current user name.\s
                newName - An updated user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
```

```
        .region(region)
        .build();

    updateIAMUser(iam, curName, newName);
    System.out.println("Done");
    iam.close();
}

public static void updateIAMUser(IamClient iam, String curName, String newName)
{
    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s", newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateUser](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Créer et gérer un service résilient

L'exemple de code suivant montre comment créer un service Web à charge équilibrée qui renvoie des recommandations de livres, de films et de chansons. L'exemple montre comment le service répond aux défaillances et comment le restructurer pour accroître la résilience en cas de défaillance.

- Utilisez un groupe Amazon EC2 Auto Scaling pour créer des instances Amazon Elastic Compute Cloud (Amazon EC2) sur la base d'un modèle de lancement et pour maintenir le nombre d'instances dans une plage spécifiée.

- Gérez et distribuez les requêtes HTTP avec Elastic Load Balancing.
- Surveillez l'état des instances d'un groupe Auto Scaling et transférez les demandes uniquement aux instances saines.
- Exécutez un serveur Web Python sur chaque instance EC2 pour gérer les requêtes HTTP. Le serveur Web répond par des recommandations et des surveillances de l'état.
- Simulez un service de recommandation avec une table Amazon DynamoDB.
- Contrôlez la réponse du serveur Web aux demandes et aux contrôles de santé en mettant à jour AWS Systems Manager les paramètres.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";
```

```
public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

public static final String targetGroupName = "doc-example-resilience-tg";
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println("""
        This concludes the demo of how to build and manage a resilient
service.
```

```
        To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
        that were created for this demo.
        """);

    System.out.println("\n Do you want to delete the resources (y/n)? ");
    String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

    if (userInput.equals("y")) {
        // Delete resources here
        deleteResources(loadBalancer, autoScaler, database);
        System.out.println("Resources deleted.");
    } else {
        System.out.println("""
            Okay, we'll leave the resources intact.
            Don't forget to delete them when you're done with them or you
might incur unexpected charges.
            """);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The example has completed. ");
    System.out.println("\n Thanks for watching!");
    System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
```

```
System.out.println(
    """
        For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
        against various kinds of failures.

        Some of the resources create by this demo are:
        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
    """);

System.out.println("Press Enter when you're ready.");
in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating and populating a DynamoDB table named " +
tableName);
Database database = new Database();
database.createTable(tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
    This script starts a Python web server defined in the `server.py`
script. The web server
    listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
    For demo purposes, this server is run as the root user. In
production, the best practice is to
    run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to
assume a role that grants
```

```
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
            defines how the load balancer connects to instances. The load
balancer provides a
```

```

        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessful) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                        For this example to work, the default security group for
your default VPC must
                        allow access from this computer. You can either add it
automatically from this
                        example or add it yourself using the AWS Management
Console.
                    """);
                }
            }
        }
    }
}

```

```
        System.out.println(
            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
        System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);
```

```
System.out.println("Resetting parameters to starting values for demo.");
paramHelper.reset();

System.out.println(
    """
        This part of the demonstration shows how to toggle
different parts of the system
        to create situations where the web service fails, and shows
how using a resilient
        architecture can keep the web service running in spite of
these failures.

        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
    """);
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    """
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
    """);
```



```
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns a
static response.
    The service still reports as healthy because health checks are still
shallow.
    """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
    + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);
```

```
demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
    is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
    instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
    the load balancer takes unhealthy instances out of its rotation.
    """);

demoChoices(loadBalancer);

System.out.println(
    """
        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
        instance is to terminate it and let the auto scaler start a
new instance to replace it.
    """);
autoScaler.terminateInstance(badInstanceId);
```

```
        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
```

```
System.out.print("\nWhich action would you like to take? ");
int choice = scanner.nextInt();
System.out.println("-".repeat(88));

switch (choice) {
    case 0 -> {
        System.out.println("Request:\n");
        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
        CloseableHttpClient httpClient =
HttpClient.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);

        // Display the JSON response
        BufferedReader reader = new BufferedReader(
            new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");
```

```

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
check to update
                Note that it can take a minute or two for the health
                after changes are made.
                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
}

```

Créez une classe qui englobe les actions Auto Scaling et Amazon EC2.

```

public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
}

```

```
private static IamClient iamClient;

private static SsmClient ssmClient;

private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
```

```
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
    .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
    ReplaceIamInstanceProfileAssociationRequest
        .builder()
        .iamInstanceProfile(iamInstanceProfile)
        .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
        .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    }
}
```

```
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }
    System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
        newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
    while (!instReady) {
        if (tries % 6 == 0) {
            getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                .instanceIds(instanceId)
                .build());
            System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
        }
        tries++;
        try {
            TimeUnit.SECONDS.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
```



```
        Collections.singletonList("cd / && sudo python3 server.py
80"))))
        .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.getInstanceProfile().getInstanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .roleName(roleName)
```

```
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
        .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(attachedPolicy.policyArn())
            .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}
```

```
}

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /**
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                .name("group-name")
                .values("default")
                .build();

            Filter filter1 = Filter.builder()
                .name("vpc-id")
                .values(VPC)
                .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                .filters(filter, filter1)
                .build();
```

```
DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
    .describeSecurityGroups(securityGroupsRequest);
String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
groupInfo.setGroupName(securityGroup);

for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
    System.out.println("Found security group: " + secGroup.groupId());

    for (IpPermission ipPermission : secGroup.ipPermissions()) {
        if (ipPermission.fromPort() == port) {
            System.out.println("Found inbound rule: " + ipPermission);
            for (IpRange ipRange : ipPermission.ipRanges()) {
                String cidrIp = ipRange.cidrIp();
                if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                    System.out.println(cidrIp + " is applicable");
                    portIsOpen = true;
                }
            }

            if (!ipPermission.prefixListIds().isEmpty()) {
                System.out.println("Prefix lList is applicable");
                portIsOpen = true;
            }

            if (!portIsOpen) {
                System.out
                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
            } else {
                break;
            }
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}
```

```
        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /**
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                .autoScalingGroupName(asGroupName)
                .targetGroupARNs(targetGroupARN)
                .build();

            getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
```

```
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
```

```
        .builder()
        .filters(defaultFilter)
        .build();

DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
```

```
AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
List<String> instanceIds = autoScalingGroup.instances().stream()
    .map(instance -> instance.instanceId())
    .collect(Collectors.toList());

String[] instanceIdArray = instanceIds.toArray(new String[0]);
for (String instanceId : instanceIdArray) {
    System.out.println("Instance ID: " + instanceId);
    return instanceId;
}
return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
```



```
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
    .listEntitiesForPolicy(listEntitiesRequest);
if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
    || !listEntitiesResponse.policyRoles().isEmpty()) {
    // Detach the policy from any entities it is attached to.
    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy.arn())
    .roleName(roleName) // Specify the name of the IAM role
    .build();

    getIAMClient().detachRolePolicy(detachPolicyRequest);
    System.out.println("Policy detached from entities.");
}

// Now, you can delete the policy.
DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

getIAMClient().deletePolicy(deletePolicyRequest);
System.out.println("Policy deleted successfully.");
break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
```

```

        RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

        getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
        System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
    }

    // Delete the instance profile after removing all roles
    DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .build();

    getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
    System.out.println(InstanceProfile + " Deleted");
    System.out.println("All roles and policies are deleted.");
}
}

```

Créez une classe qui englobe les actions Elastic Load Balancing.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {

```

```
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
    .names(targetGroupName)
    .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
    .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
    .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                .build();

            getLoadBalancerClient().deleteLoadBalancer(
                builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                .waitUntilLoadBalancersDeleted(request);
```

```
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
            }
        }
    }
}
```

```
        System.out.println("Got connection error from load balancer
endpoint, retrying...");
        TimeUnit.SECONDS.sleep(15);
    }
}

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
```

```
* Creates an Elastic Load Balancing load balancer that uses the specified
* subnets
* and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
```

```
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Créez une classe qui utilise DynamoDB pour simuler un service de recommandation.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }
}
```

```
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()

```



```

        .attributeName("ItemId")
        .attributeType(ScalarAttributeType.N)
        .build())
    .keySchema(
        KeySchemaElement.builder()
            .attributeName("MediaType")
            .keyType(KeyType.HASH)
            .build(),
        KeySchemaElement.builder()
            .attributeName("ItemId")
            .keyType(KeyType.RANGE)
            .build())
    .provisionedThroughput(
        ProvisionedThroughput.builder()
            .readCapacityUnits(5L)
            .writeCapacityUnits(5L)
            .build())
    .build();

    getDynamoDbClient().createTable(createTableRequest);
    System.out.println("Creating table " + tableName + "...");

    // Wait until the Amazon DynamoDB table is created.
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

    // Add records to the table.
    populateTable(fileName, tableName);
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.

```

```

public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}

```

Créez une classe qui englobe les actions de Systems Manager.

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
    }
}

```

```
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();


        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)

- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Créer un utilisateur et assumer d'un rôle

L'exemple de code suivant montre comment créer un utilisateur et assumer un rôle.

 Warning

Afin d'éviter les risques de sécurité, n'employez pas les utilisateurs IAM pour l'authentification lorsque vous développez des logiciels spécialisés ou lorsque vous travaillez avec des données réelles. Préférez la fédération avec un fournisseur d'identité tel que [AWS IAM Identity Center](#).

- Créer un utilisateur sans autorisation.
- Créer un rôle qui accorde l'autorisation de répertorier les compartiments Amazon S3 pour le compte.
- Ajouter une politique pour permettre à l'utilisateur d'assumer le rôle.
- Assumez le rôle et répertorier les compartiments S3 à l'aide d'informations d'identification temporaires, puis nettoyez les ressources.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez des fonctions qui encapsulent les actions de l'utilisateur IAM.

```
/*
  To run this Java V2 code example, set up your development environment, including
  your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3 Service
  client object with the temporary credentials.
  6. Deletes the resources.
*/

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\"," +
        "  \"Statement\": [" +
        "    {" +
        "      \"Effect\": \"Allow\"," +
        "      \"Action\": [" +
        "        \"s3:*\"" +
        "      ]," +
        "      \"Resource\": \"*\"" +
        "    }" +
        "  ]" +
        "}";
```

```
public static String userArn;

public static void main(String[] args) throws Exception {

    final String usage = ""

        Usage:
            <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

        Where:
            username - The name of the IAM user to create.\s
            policyName - The name of the policy to create.\s
            roleName - The name of the role to create.\s
            roleSessionName - The name of the session required for the
assumeRole operation.\s
            bucketName - The name of the Amazon S3 bucket from which objects
are read.\s

        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    String policyName = args[1];
    String roleName = args[2];
    String roleSessionName = args[3];
    String bucketName = args[4];

    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS IAM example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(" 1. Create the IAM user.");
    User createUser = createIAMUser(iam, userName);
```

```
System.out.println(DASHES);
userArn = createUser.arn();

AccessKey myKey = createIAMAccessKey(iam, userName);
String accessKey = myKey.accessKeyId();
String secretKey = myKey.secretAccessKey();
String assumeRolePolicyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
    "\"Statement\": [{" +
    "\"Effect\": \"Allow\"," +
    "\"Principal\": {" +
    "  \"AWS\": \"" + userArn + "\"" +
    "}," +
    "\"Action\": \"sts:AssumeRole\"" +
    "}]}" +
    "}";

System.out.println(assumeRolePolicyDocument);
System.out.println(userName + " was successfully created.");
System.out.println(DASHES);
System.out.println("2. Creates a policy.");
String polArn = createIAMPolicy(iam, policyName);
System.out.println("The policy " + polArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Creates a role.");
TimeUnit.SECONDS.sleep(30);
String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
System.out.println(roleArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Grants the user permissions.");
attachIAMRolePolicy(iam, roleName, polArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait for 30 secs so the resource is available");
TimeUnit.SECONDS.sleep(30);
System.out.println("5. Gets temporary credentials by assuming the role.");
System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
```

```
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user)
                .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                .userName(username)
                .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                .userName(response.user().userName())
                .build();
```



```
        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static String createIAMRole(IamClient iam, String rolename, String json)
{

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " + response.role().arn());
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
```

```
        .policyArn(response.policy().arn())
        .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
        String polArn;
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.attachRolePolicy(attachRequest);
    }
}
```

```
        System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
    String keySecret) {

    // Use the creds of the new IAM user that was created in this code example.
    AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(StaticCredentialsProvider.create(credentials))
        .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
        String secToken = myCreds.sessionToken();

        // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
        // invoking assumeRole.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .credentialsProvider(
StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
            .region(region)
```

```
        .build();

        System.out.println("Created a S3Client using temp credentials.");
        System.out.println("Listing objects in " + bucketName);
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.println("The name of the key is " + myValue.key());
            System.out.println("The owner is " + myValue.owner());
        }

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {

    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted " + polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
```

```
        .build());

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

Utiliser l'API IAM Policy Builder

L'exemple de code suivant illustre comment :

- Créez des politiques IAM à l'aide de l'API orientée objet.
- Utilisez l'API IAM Policy Builder avec le service IAM.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Les exemples utilisent les importations suivantes.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Créez une politique basée sur le temps.

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1

        .operator(IamConditionOperator.DATE_GREATER_THAN)

        .key("aws:CurrentTime")

        .value("2020-04-01T00:00:00Z"))
            .addCondition(b1 -> b1

        .operator(IamConditionOperator.DATE_LESS_THAN)

        .key("aws:CurrentTime")
```

```

        .value("2020-06-30T23:59:59Z"))
            .build());

    // Use an IamPolicyWriter to write out the JSON string to a more
readable
    // format.
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
        .build());
}

```

Créez une politique comportant plusieurs conditions.

```

public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
            .addAction("dynamodb:Query")
            .addAction("dynamodb:PutItem")
            .addAction("dynamodb:UpdateItem")
            .addAction("dynamodb:DeleteItem")

            .addAction("dynamodb:BatchWriteItem")

            .addResource("arn:aws:dynamodb:*:*:table/table-name")

            .addConditions(IamConditionOperator.STRING_EQUALS

            .addPrefix("ForAllValues:"),

            "dynamodb:Attributes",

            List.of("column-
name1", "column-name2", "column-name3"))

            .addCondition(b1 -> b1

            .operator(IamConditionOperator.STRING_EQUALS

            .addSuffix("IfExists"))
}

```



```

    .key("dynamodb:Select")

    .value("SPECIFIC_ATTRIBUTES"))))
        .build();

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Utilisez des principaux dans une politique.

```

public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)

        .addResource("arn:aws:s3:::BUCKETNAME/*")

        .addResource("arn:aws:s3:::BUCKETNAME")
            .addCondition(b1 -> b1

        .operator(IamConditionOperator.ARN_NOT_EQUALS)

        .key("aws:PrincipalArn")

        .value("arn:aws:iam::444455556666:user/user-name"))))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Autorisez l'accès intercompte .

```

public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)

```

```

        .addPrincipal(IamPrincipalType.AWS,
"111122223333")
        .addAction("s3:PutObject")
        .addResource("arn:aws:s3:::DOC-
EXAMPLE-BUCKET/*")
        .addCondition(b1 -> b1
        .operator(IamConditionOperator.STRING_EQUALS)
        .key("s3:x-amz-acl")
        .value("bucket-
owner-full-control"))))
        .build();
return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Créez et chargez une IamPolicy.

```

public String createAndUploadPolicyExample(IamClient iam, String accountID,
String policyName) {
    // Build the policy.
    IamPolicy policy = IamPolicy.builder() // 'version' defaults to
"2012-10-17".
        .addStatement(IamStatement.builder()
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:PutItem")
            .addResource("arn:aws:dynamodb:us-
east-1:" + accountID
                + ":table/
exampleTableName")
            .build())
        .build();
    // Upload the policy.
    iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
    return
policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

Téléchargez et utilisez une IamPolicy.

```
public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
            String newPolicyName) {

    String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;

    GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

    String policyVersion =
getPolicyResponse.policy().defaultVersionId();
    GetPolicyVersionResponse getPolicyVersionResponse = iam
        .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

    // Create an IamPolicy instance from the JSON string returned from
IAM.

    String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
        StandardCharsets.UTF_8);
    IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

    /*
    * All IamPolicy components are immutable, so use the copy method
that creates a
    * new instance that
    * can be altered in the same method call.
    *
    * Add the ability to get an item from DynamoDB as an additional
action.
    */
    IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

    // Create a new statement that replaces the original statement.
    IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

    // Upload the new policy. IAM now has both policies.
    iam.createPolicy(r -> r.policyName(newPolicyName)
        .policyDocument(newPolicy.toJson()));
}
```

```
        return
        newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }
```

- Pour de plus amples informations, consultez le [Guide du développeur AWS SDK for Java 2.x](#).
- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreatePolicy](#)
 - [GetPolicy](#)
 - [GetPolicyVersion](#)

AWS IoT exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with AWS IoT.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour AWS IoT

Les exemples de code suivants montrent comment démarrer avec AWS IoT.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings( IotClient iotClient) {
        ListThingsRequest thingsRequest = ListThingsRequest.builder()
            .maxResults(10)
            .build();

        ListThingsResponse response = iotClient.listThings(thingsRequest) ;
        List<ThingAttribute> thingList = response.things();
        for (ThingAttribute attribute : thingList) {
            System.out.println("Thing name: "+attribute.thingName());
            System.out.println("Thing ARN: "+attribute.thingArn());
        }
    }
}
```

- Pour plus de détails sur l'API, voir [ListThings](#) dans le manuel de référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

AttachThingPrincipal

L'exemple de code suivant montre comment utiliser `AttachThingPrincipal`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
    .thingName(thingName)
    .principal(certificateArn)
    .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
    if (attachResponse.sdkHttpResponse().isSuccessful()) {
        System.out.println("Certificate attached to Thing successfully.");

        // Print additional information about the Thing.
        describeThing(iotClient, thingName);
    } else {
```

```
        System.err.println("Failed to attach certificate to Thing. HTTP Status  
Code: " +  
        attachResponse.sdkHttpResponse().statusCode());  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [AttachThingPrincipal](#) à la section Référence des AWS SDK for Java 2.x API.

CreateKeysAndCertificate

L'exemple de code suivant montre comment utiliser `CreateKeysAndCertificate`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createCertificate(IotClient iotClient) {  
    try {  
        CreateKeysAndCertificateResponse response =  
        iotClient.createKeysAndCertificate();  
        String certificatePem = response.certificatePem();  
        String certificateArn = response.certificateArn();  
  
        // Print the details.  
        System.out.println("\nCertificate:");  
        System.out.println(certificatePem);  
        System.out.println("\nCertificate ARN:");  
        System.out.println(certificateArn);  
        return certificateArn;  
  
    } catch (IotException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
        return "";  
    }
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeysAndCertificate](#) à la section Référence des AWS SDK for Java 2.x API.

CreateThing

L'exemple de code suivant montre comment utiliser `CreateThing`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createIoTThing(IotClient iotClient, String thingName) {  
    try {  
        CreateThingRequest createThingRequest = CreateThingRequest.builder()  
            .thingName(thingName)  
            .build();  
  
        CreateThingResponse createThingResponse =  
            iotClient.createThing(createThingRequest);  
        System.out.println(thingName + " was successfully created. The ARN value  
is " + createThingResponse.thingArn());  
  
    } catch (IotException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateThing](#) à la section Référence des AWS SDK for Java 2.x API.

CreateTopicRule

L'exemple de code suivant montre comment utiliser `CreateTopicRule`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");
    }
}
```

```
    } catch (IotException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTopicRule](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteCertificate

L'exemple de code suivant montre comment utiliser `DeleteCertificate`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteCertificate(IotClient iotClient, String  
certificateArn ) {  
    DeleteCertificateRequest certificateProviderRequest =  
DeleteCertificateRequest.builder()  
        .certificateId(extractCertificateId(certificateArn))  
        .build();  
  
    iotClient.deleteCertificate(certificateProviderRequest);  
    System.out.println(certificateArn + " was successfully deleted.");  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCertificate](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteThing

L'exemple de code suivant montre comment utiliser `DeleteThing`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteThing](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeEndpoint

L'exemple de code suivant montre comment utiliser `DescribeEndpoint`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String describeEndpoint(IotClient iotClient) {
```

```
    try {
        DescribeEndpointResponse endpointResponse =
iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeEndpoint](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeThing

L'exemple de code suivant montre comment utiliser `DescribeThing`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build() ;
    }
```

```
        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeThing](#) à la section Référence des AWS SDK for Java 2.x API.

DetachThingPrincipal

L'exemple de code suivant montre comment utiliser `DetachThingPrincipal`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
    }
```

```
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetachThingPrincipal](#) à la section Référence des AWS SDK for Java 2.x API.

ListCertificates

L'exemple de code suivant montre comment utiliser `ListCertificates`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListCertificates](#) à la section Référence des AWS SDK for Java 2.x API.

SearchIndex

L'exemple de code suivant montre comment utiliser `SearchIndex`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
        iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
            System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SearchIndex](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateThing

L'exemple de code suivant montre comment utiliser `UpdateThing`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```


- Pour plus de détails sur l'API, reportez-vous [UpdateThing](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Travaillez avec des cas d'utilisation liés à la gestion des appareils

L'exemple de code suivant montre comment utiliser les cas d'utilisation de la gestion des AWS IoT appareils à l'aide du AWS IoT SDK

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.AttributePayload;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
```

```
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach and delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
private static final String TOPIC = "your-iot-topic";
public static void main(String[] args) {
    final String usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
            snsAction - An ARN of an SNS topic.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String thingName;
    String ruleName;
    String roleARN = args[0];
    String snsAction = args[1];
    Scanner scanner = new Scanner(System.in);
    IotClient iotClient = IotClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS IoT example workflow.");
    System.out.println("""
        This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series of
steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality for
creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.
        """);
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
}
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an AWS IoT Thing.");
System.out.println("""
    An AWS IoT Thing represents a virtual entity in the AWS IoT service that
    can be associated with a physical device.
    """);
// Prompt the user for input.
System.out.print("Enter Thing name: ");
thingName = scanner.nextLine();
createIoTThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Generate a device certificate.");
System.out.println("""
    A device certificate performs a role in securing the communication
    between devices (Things) and the AWS IoT platform.
    """);

System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
String certAns = scanner.nextLine();
String certificateArn="" ;
if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
    certificateArn = createCertificate(iotClient);
    System.out.println("Attach the certificate to the AWS IoT Thing.");
    attachCertificateToThing(iotClient, thingName, certificateArn);
} else {
    System.out.println("A device certificate was not created.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Update an AWS IoT Thing with Attributes.");
System.out.println("""
    IoT Thing attributes, represented as key-value pairs, offer a pivotal
    advantage in facilitating efficient data
    management and retrieval within the AWS IoT ecosystem.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
updateThing(iotClient, thingName);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
System.out.println("""
    An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
String endpointUrl = describeEndpoint(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List your AWS IoT certificates");
System.out.print("Press Enter to continue...");
scanner.nextLine();
if (certificateArn.length() > 0) {
    listCertificates(iotClient);
} else {
    System.out.println("You did not create a certificates. Skipping this
step.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
System.out.println("""
    A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
    of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
    the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
    .region(Region.US_EAST_1)
    .endpointOverride(URI.create(endpointUrl))
    .build();
```

```
updateShadowThing(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Write out the state information, in JSON format.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
getPayload(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Creates a rule");
System.out.println("""
Creates a rule that is an administrator-level action.
Any user who has permission to create rules will be able to access data
processed by the rule.
""");
System.out.print("Enter Rule name: ");
ruleName = scanner.nextLine();
createIoTRule(iotClient, roleARN, ruleName, snsAction);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List your rules.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
listIoTRules(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Search things using the Thing name.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
String queryString = "thingName:"+thingName ;
searchThings(iotClient, queryString);
System.out.println(DASHES);

System.out.println(DASHES);
if (certificateArn.length() > 0) {
    System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
```

```

        System.out.println("11. You selected to detach amd delete the
certificate.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        detachThingPrincipal(iotClient, thingName, certificateArn);
        deleteCertificate(iotClient, certificateArn);
    } else {
        System.out.println("11. You selected not to delete the
certificate.");
    }
    } else {
        System.out.println("11. You did not create a certificate so there is
nothing to delete.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("12. Delete the AWS IoT Thing.");
    System.out.print("Do you want to delete the IoT Thing? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        deleteIoTThing(iotClient, thingName);
    } else {
        System.out.println("The IoT Thing was not deleted.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The AWS IoT workflow has successfully completed.");
    System.out.println(DASHES);
}

public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}

public static void listIoTRules(IotClient iotClient) {
    try {

```

```
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        ListTopicRulesResponse listTopicRulesResponse =
iotClient.listTopicRules(listTopicRulesRequest);
        System.out.println("List of IoT Rules:");
        List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
        for (TopicRuleListItem rule : ruleList) {
            System.out.println("Rule Name: " + rule.ruleName());
            System.out.println("Rule ARN: " + rule.ruleArn());
            System.out.println("-----");
        }

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();
```



```
        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}\";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
```

```
        .payload(data)
        .build();

    // Update Thing Shadow.
    iotPlaneClient.updateThingShadow(updateThingShadowRequest);
    System.out.println("Thing Shadow updated successfully.");

} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String describeEndpoint(IotClient iotClient) {
    try {
```

```
        DescribeEndpointResponse endpointResponse =
iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}

public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();
```

```
        iotClient.deleteCertificate(certificateProviderRequest);
        System.out.println(certificateArn + " was successfully deleted.");
    }

    // Get the cert Id from the Cert ARN value.
    private static String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
    }

    public static String createCertificate(IotClient iotClient) {
        try {
            CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
            String certificatePem = response.certificatePem();
            String certificateArn = response.certificateArn();

            // Print the details.
            System.out.println("\nCertificate:");
            System.out.println(certificatePem);
            System.out.println("\nCertificate ARN:");
            System.out.println(certificateArn);
            return certificateArn;

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }

    public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
        // Attach the certificate to the thing.
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();
```

```
AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

// Verify the attachment was successful.
if (attachResponse.sdkHttpResponse().isSuccessful()) {
    System.out.println("Certificate attached to Thing successfully.");

    // Print additional information about the Thing.
    describeThing(iotClient, thingName);
} else {
    System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
        attachResponse.sdkHttpResponse().statusCode());
}
}

private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);
    }
}
```

```
        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createIoTThing(IotClient iotClient, String thingName) {
        try {
            CreateThingRequest createThingRequest = CreateThingRequest.builder()
                .thingName(thingName)
                .build();

            CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
            System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*)\\.iot\\.us-east-1\\.amazonaws\\.
\\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    public static void searchThings(IotClient iotClient, String queryString){
```

```
SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
    .queryString(queryString)
    .build();

try {
    // Perform the search and get the result.
    SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

    // Process the result.
    if (searchIndexResponse.things().isEmpty()) {
        System.out.println("No things found.");
    } else {
        searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
    }
} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

AWS IoT data exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with AWS IoT data.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

GetThingShadow

L'exemple de code suivant montre comment utiliser `GetThingShadow`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetThingShadow](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateThingShadow

L'exemple de code suivant montre comment utiliser `UpdateThingShadow`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
        \"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateThingShadow](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'Amazon Keyspaces utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Keyspaces.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Amazon Keyspaces

Les exemples de code suivants montrent comment commencer à utiliser Amazon Keyspaces.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                .maxResults(10)
                .build();

            ListKeyspacesResponse response =
keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace : keyspaces) {
                System.out.println("The name of the keyspace is " +
keyspace.keyspaceName());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyspaces](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateKeyspace

L'exemple de code suivant montre comment utiliser `CreateKeyspace`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyspace](#) à la section Référence des AWS SDK for Java 2.x API.

CreateTable

L'exemple de code suivant montre comment utiliser `CreateTable`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();
```

```
PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());


} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTable](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteKeyspace

L'exemple de code suivant montre comment utiliser `DeleteKeyspace`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);


    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyspace](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteTable

L'exemple de code suivant montre comment utiliser `DeleteTable`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTable](#) à la section Référence des AWS SDK for Java 2.x API.

GetKeyspace

L'exemple de code suivant montre comment utiliser `GetKeyspace`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
    }
}
```



```
        System.out.println("The " + name + " KeySpace is ready");
    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetKeyspace](#) à la section Référence des AWS SDK for Java 2.x API.

GetTable

L'exemple de code suivant montre comment utiliser `GetTable`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void checkTable(KeyspacesClient keyClient, String keySpaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keySpaceName(keySpaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);
        }
    }
}
```

```
        if (status.compareTo("ACTIVE") == 0) {
            tableStatus = true;
        }
        Thread.sleep(500);
    }

    List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
    for (ColumnDefinition def : cols) {
        System.out.println("The column name is " + def.name());
        System.out.println("The column type is " + def.type());
    }

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetTable](#) à la section Référence des AWS SDK for Java 2.x API.

ListKeyspaces

L'exemple de code suivant montre comment utiliser `ListKeyspaces`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();
```

```
        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyspaces](#) à la section Référence des AWS SDK for Java 2.x API.

ListTables

L'exemple de code suivant montre comment utiliser `ListTables`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
```

```
        " Table name: " + content.tableName()));  
  
    } catch (KeyspacesException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTables](#) à la section Référence des AWS SDK for Java 2.x API.

RestoreTable

L'exemple de code suivant montre comment utiliser `RestoreTable`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,  
    ZonedDateTime utc) {  
    try {  
        Instant myTime = utc.toInstant();  
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()  
            .restoreTimestamp(myTime)  
            .sourceTableName("Movie")  
            .targetKeyspaceName(keyspaceName)  
            .targetTableName("MovieRestore")  
            .sourceKeyspaceName(keyspaceName)  
            .build();  
  
        RestoreTableResponse response =  
            keyClient.restoreTable(restoreTableRequest);  
        System.out.println("The ARN of the restored table is " +  
            response.restoredTableARN());  
    }  
}
```

```
    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [RestoreTable](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateTable

L'exemple de code suivant montre comment utiliser `UpdateTable`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateTable](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Commencez avec les espaces de touches et les tableaux

L'exemple de code suivant illustre comment :

- Créez un espace de touches et un tableau. Le schéma de table contient les données vidéo et la point-in-time restauration est activée.
- Connectez-vous au keyspace à l'aide d'une connexion TLS sécurisée avec authentification SigV4.
- Interrogez la table. Ajoutez, récupérez et mettez à jour les données des films.
- Mettez à jour le tableau. Ajoutez une colonne pour suivre les films visionnés.
- Restaurez l'état précédent de la table et nettoyez les ressources.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**  
 * Before running this Java (v2) code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * Before running this Java code example, you must create a  
 * Java keystore (JKS) file and place it in your project's resources folder.  
 *
```

```

* This file is a secure file format used to hold certificate information for
* Java applications. This is required to make a connection to Amazon Keyspaces.
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/keyspaces/latest/devguide/using\_java\_driver.html
*
* This Java example performs the following tasks:
*
* 1. Create a keyspace.
* 2. Check for keyspace existence.
* 3. List keyspaces using a paginator.
* 4. Create a table with a simple movie data schema and enable point-in-time
* recovery.
* 5. Check for the table to be in an Active state.
* 6. List all tables in the keyspace.
* 7. Use a Cassandra driver to insert some records into the Movie table.
* 8. Get all records from the Movie table.
* 9. Get a specific Movie.
* 10. Get a UTC timestamp for the current time.
* 11. Update the table schema to add a 'watched' Boolean column.
* 12. Update an item as watched.
* 13. Query for items with watched = True.
* 14. Restore the table back to the previous state using the timestamp.
* 15. Check for completion of the restore action.
* 16. Delete the table.
* 17. Confirm that both tables are deleted.
* 18. Delete the keyspace.
*/

```

```

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException, IOException
    {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
    }
}

```

```
String tableName = "Movie";
String tableNameRestore = "MovieRestore";
Region region = Region.US_EAST_1;
KeyspacesClient keyClient = KeyspacesClient.builder()
    .region(region)
    .build();

DriverConfigLoader loader =
DriverConfigLoader.fromClasspath("application.conf");
CqlSession session = CqlSession.builder()
    .withConfigLoader(loader)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Keyspaces example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a keyspace.");
createKeyspace(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
Thread.sleep(5000);
System.out.println("2. Check for keyspace existence.");
checkKeyspaceExistence(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List keyspaces using a paginator.");
listKeyspacesPaginator(keyClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
createTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Check for the table to be in an Active state.");
Thread.sleep(6000);
checkTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);
```



```
System.out.println(DASHES);
System.out.println("6. List all tables in the keyspace.");
listTables(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
Thread.sleep(6000);
loadData(session, fileName, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get all records from the Movie table.");
getMovieData(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get a specific Movie.");
getSpecificMovie(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a UTC timestamp for the current time.");
ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
System.out.println("DATETIME = " + Date.from(utc.toInstant()));
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Update the table schema to add a watched Boolean
column.");
updateTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Update an item as watched.");
Thread.sleep(10000); // Wait 10 secs for the update.
updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Query for items with watched = True.");
getWatchedData(session, keyspaceName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state using
the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
```

```
        .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
    try {
        String status;
        GetTableResponse response;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        // Keep looping until table cannot be found and a
ResourceNotFoundException is
        // thrown.
        while (true) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);
            Thread.sleep(500);
        }

    } catch (ResourceNotFoundException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println("The table is deleted");
}

public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();
```

```
        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println("The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```

    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                .restoreTimestamp(myTime)
                .sourceTableName("Movie")
                .targetKeyspaceName(keyspaceName)
                .targetTableName("MovieRestore")
                .sourceKeyspaceName(keyspaceName)
                .build();

            RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
            response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session
            .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
            + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
        BatchStatementBuilder builder =
        BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", titleUpdate)

```

```
        .setInt("k1", yearUpdate)
        .build());

    BatchStatement batchStatement = builder.build();
    session.execute(batchStatement);
}

public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificMovie(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute(
        "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
Family' ALLOW FILTERING ;");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

// Get records from the Movie table.
public static void getMovieData(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
    "\".\"Movie\";");
}
```

```
resultSet.forEach(item -> {
    System.out.println("The Movie title is " + item.getString("title"));
    System.out.println("The Movie year is " + item.getInt("year"));
    System.out.println("The plot is " + item.getString("plot"));
});
}

// Load data into the table.
public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
    String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {

        // Add 20 movies to the table.
        if (t == 20)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String plot = currentNode.path("info").path("plot").toString();

        // Insert the data into the Amazon Keyspaces table.
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", title)
            .setInt("k1", year)
            .setString("k2", plot)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
        t++;
    }
}
```

```
        System.out.println("You have added " + t + " records successfully!");
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                .flatMap(r -> r.tables().stream())
                .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                    " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
        throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
            }
        }
    }
}
```



```
        Thread.sleep(500);
    }

    List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
    for (ColumnDefinition def : cols) {
        System.out.println("The column name is " + def.name());
        System.out.println("The column type is " + def.type());
    }

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);
    }
}
```

```
// Set the keys.
PartitionKey yearKey = PartitionKey.builder()
    .name("year")
    .build();

PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();
```

```
        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)
 - [ListKeyspaces](#)
 - [ListTables](#)
 - [RestoreTable](#)
 - [UpdateTable](#)

Exemples Kinesis utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide de Winesis.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)
- [Exemples sans serveur](#)

Actions

CreateStream

L'exemple de code suivant montre comment utiliser `CreateStream`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
                StockTradeStream).
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String streamName = args[0];
    Region region = Region.US_EAST_1;
    KinesisClient kinesisClient = KinesisClient.builder()
        .region(region)
        .build();
    createStream(kinesisClient, streamName);
    System.out.println("Done");
    kinesisClient.close();
}

public static void createStream(KinesisClient kinesisClient, String streamName)
{
    try {
        CreateStreamRequest streamReq = CreateStreamRequest.builder()
            .streamName(streamName)
            .shardCount(1)
            .build();

        kinesisClient.createStream(streamReq);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateStream](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteStream

L'exemple de code suivant montre comment utiliser `DeleteStream`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataStream {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
                StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
```

```
        .region(region)
        .build();

    deleteStream(kinesisClient, streamName);
    kinesisClient.close();
    System.out.println("Done");
}

public static void deleteStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DeleteStreamRequest delStream = DeleteStreamRequest.builder()
            .streamName(streamName)
            .build();

        kinesisClient.deleteStream(delStream);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteStream](#) à la section Référence des AWS SDK for Java 2.x API.

GetRecords

L'exemple de code suivant montre comment utiliser `GetRecords`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
```



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetRecords {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to read from (for
example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();
    }
}
```

```
        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
        do {
            streamRes = kinesisClient.describeStream(describeStreamRequest);
            shards.addAll(streamRes.streamDescription().shards());

            if (shards.size() > 0) {
                lastShardId = shards.get(shards.size() - 1).shardId();
            }
        } while (streamRes.streamDescription().hasMoreShards());

        GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
            .streamName(streamName)
            .shardIteratorType("TRIM_HORIZON")
            .shardId(lastShardId)
            .build();

        GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
        shardIterator = shardIteratorResult.shardIterator();

        // Continuously read data records from shard.
        List<Record> records;

        // Create new GetRecordsRequest with existing shardIterator.
        // Set maximum records to return to 1000.
        GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
            .shardIterator(shardIterator)
            .limit(1000)
            .build();
```

```
GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

// Put result into record list. Result may be empty.
records = result.records();

// Print records
for (Record record : records) {
    SdkBytes byteBuffer = record.data();
    System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetRecords](#) à la section Référence des AWS SDK for Java 2.x API.

PutRecord

L'exemple de code suivant montre comment utiliser PutRecord.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();

        // Ensure that the Kinesis Stream is valid.
        validateStream(kinesisClient, streamName);
        setStockData(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void setStockData(KinesisClient kinesisClient, String streamName)
    {
        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x = 0; x < index; x++) {
                StockTrade trade = stockTradeGenerator.getRandomTrade();
            }
        }
    }
}
```

```

        sendStockTrade(trade, kinesisClient, streamName);
        Thread.sleep(100);
    }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient
kinesisClient,
    String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization
    by
    // the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol
    as the partition key, explained in
                                                // the Supplemental
    Information section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        System.err.println(e.getMessage());
    }
}

private static void validateStream(KinesisClient kinesisClient, String
streamName) {
    try {

```

```
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
        .streamName(streamName)
        .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }

        } catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutRecord](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples sans serveur

Invoker une fonction Lambda à partir d'un déclencheur Kinesis

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements provenant d'un flux Kinesis. La fonction récupère la charge utile Kinesis, décode à partir de Base64 et enregistre le contenu de l'enregistrement.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Consommation d'un événement Kinesis avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }
}
```

```
}
```

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur Kinesis

L'exemple de code suivant montre comment implémenter une réponse par lots partielle pour les fonctions Lambda qui reçoivent des événements d'un flux Kinesis. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Signalement des échecs d'articles par lots Kinesis avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";
```



```
        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse(batchItemFailures);
    }
}
```

AWS KMS exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with AWS KMS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Clé Hello KMS

L'exemple de code suivant montre comment commencer à utiliser la clé KMS.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();
        }
    }
}
```

```
        ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
        keysResponse.stream()
            .flatMap(r -> r.keys().stream())
            .forEach(key -> System.out
                .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [listKeysPaginator](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateAlias

L'exemple de code suivant montre comment utiliser `CreateAlias`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
```

```
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);
        System.out.println(aliasName + " was successfully created.");

    } catch (ResourceExistsException e) {
        System.err.println("Alias already exists: " + e.getMessage());
        System.err.println("Moving on...");
    } catch (Exception e) {
        System.err.println("An unexpected error occurred: " + e.getMessage());
        System.err.println("Moving on...");
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateAlias](#) à la section Référence des AWS SDK for Java 2.x API.

CreateGrant

L'exemple de code suivant montre comment utiliser `CreateGrant`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);
```

```
        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateGrant](#) à la section Référence des AWS SDK for Java 2.x API.

CreateKey

L'exemple de code suivant montre comment utiliser `CreateKey`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();
```

```
        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKey](#) à la section Référence des AWS SDK for Java 2.x API.

Decrypt

L'exemple de code suivant montre comment utiliser Decrypt.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

    } catch (KmsException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, voir [Déchiffrer](#) dans le guide de référence des AWS SDK for Java 2.x API.

DeleteAlias

L'exemple de code suivant montre comment utiliser `DeleteAlias`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAlias](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeKey

L'exemple de code suivant montre comment utiliser `DescribeKey`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
            System.out.println("The key is not enabled. Key state: " +
keyState);
        }

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeKey](#) à la section Référence des AWS SDK for Java 2.x API.

DisableKey

L'exemple de code suivant montre comment utiliser `DisableKey`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void disableKey(KmsClient kmsClient, String keyId) {
    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DisableKey](#) à la section Référence des AWS SDK for Java 2.x API.

EnableKey

L'exemple de code suivant montre comment utiliser `EnableKey`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Enable the KMS key.
public static void enableKey(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [EnableKey](#) à la section Référence des AWS SDK for Java 2.x API.

Encrypt

L'exemple de code suivant montre comment utiliser `Encrypt`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
    try {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
    }
}
```

```
String algorithm = response.encryptionAlgorithm().toString();
System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

// Get the encrypted data.
SdkBytes encryptedData = response.ciphertextBlob();
return encryptedData;

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return null;
}
```

- Pour plus de détails sur l'API, voir [Encrypt](#) in AWS SDK for Java 2.x API Reference.

ListAliases

L'exemple de code suivant montre comment utiliser `ListAliases`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listAllAliases(KmsClient kmsClient) {
    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
aliasesResponse.stream()
        .flatMap(r -> r.aliases().stream())
        .forEach(alias -> System.out
            .println("The alias name is: " + alias.aliasName()));
    }
}
```

```
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [ListAliases](#) à la section Référence des AWS SDK for Java 2.x API.

ListGrants

L'exemple de code suivant montre comment utiliser `ListGrants`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void displayGrantIds(KmsClient kmsClient, String keyId) {  
    try {  
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()  
            .keyId(keyId)  
            .limit(15)  
            .build();  
  
        ListGrantsIterable response =  
kmsClient.listGrantsPaginator(grantsRequest);  
        response.stream()  
            .flatMap(r -> r.grants().stream())  
            .forEach(grant -> {  
                System.out.println("The grant Id is : " + grant.grantId());  
                List<GrantOperation> ops = grant.operations();  
                for (GrantOperation op : ops) {  
                    System.out.println(op.name());  
                }  
            });  
    }  
}
```

```
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListGrants](#) à la section Référence des AWS SDK for Java 2.x API.

ListKeyPolicies

L'exemple de code suivant montre comment utiliser `ListKeyPolicies`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyPolicies](#) à la section Référence des AWS SDK for Java 2.x API.

ListKeys

L'exemple de code suivant montre comment utiliser `ListKeys`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }
}
```

```
}

public static void listAllKeys(KmsClient kmsClient) {
    try {
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
        keysResponse.stream()
            .flatMap(r -> r.keys().stream())
            .forEach(key -> System.out
                .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeys](#) à la section Référence des AWS SDK for Java 2.x API.

RevokeGrant

L'exemple de code suivant montre comment utiliser `RevokeGrant`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
```

```
try {
    RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
        .keyId(keyId)
        .grantId(grantId)
        .build();

    kmsClient.revokeGrant(grantRequest);
    System.out.println("Grant ID: [" + grantId + "] was successfully
revoked!");
} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

- Pour plus de détails sur l'API, reportez-vous [RevokeGrant](#) à la section Référence des AWS SDK for Java 2.x API.

ScheduleKeyDeletion

L'exemple de code suivant montre comment utiliser `ScheduleKeyDeletion`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        kmsClient.scheduleKeyDeletion(deletionRequest);
    }
}
```



```
        System.out.println("The key will be deleted in 7 days.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ScheduleKeyDeletion](#) à la section Référence des AWS SDK for Java 2.x API.

Sign

L'exemple de code suivant montre comment utiliser `Sign`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void signVerifyData(KmsClient kmsClient) {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest request = CreateKeyRequest.builder()
        .KeySpec(KeySpec.RSA_2048) // Specify key spec
        .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

    CreateKeyResponse response = kmsClient.createKey(request);
    String keyId2 = response.keyMetadata().keyId();
    System.out.println("Created KMS key with ID: " + keyId2);

    SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
    SignRequest signRequest = SignRequest.builder()
        .keyId(keyId2)
```

```
        .message(bytes)
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();

    SignResponse signResponse = kmsClient.sign(signRequest);
    byte[] signedBytes = signResponse.signature().asByteArray();

    // Verify the digital signature.
    VerifyRequest verifyRequest = VerifyRequest.builder()
        .keyId(keyId2)

        .message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
        .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();

    VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
    System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
    }
```

- Pour plus de détails sur l'API, consultez la section Référence de AWS SDK for Java 2.x l'API de [connexion](#).

TagResource

L'exemple de code suivant montre comment utiliser `TagResource`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
```

```
        .tagValue("Production")
        .build();

    TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
        .keyId(keyId)
        .tags(tag)
        .build();

    kmsClient.tagResource(tagResourceRequest);
    System.out.println("The key has been tagged.");

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [TagResource](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Découvrez les principales opérations de KMS

L'exemple de code suivant illustre comment :

- Créer une clé KMS.
- Répertoriez les clés KMS de votre compte et obtenez des informations les concernant.
- Activez et désactivez les clés KMS.
- Générez une clé de données symétrique qui peut être utilisée pour le chiffrement côté client.
- Générez une clé asymétrique utilisée pour signer numériquement les données.
- Clés de tag.
- Supprimez les clés KMS.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.AliasListEntry;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;
import software.amazon.awssdk.services.kms.model.DeleteAliasRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationRequest;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyResponse;
import software.amazon.awssdk.services.kms.model.GrantOperation;
import software.amazon.awssdk.services.kms.model.KeySpec;
import software.amazon.awssdk.services.kms.model.KeyState;
import software.amazon.awssdk.services.kms.model.KeyUsageType;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.LimitExceededException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesResponse;
import software.amazon.awssdk.services.kms.model.OriginType;
```

```
import software.amazon.awssdk.services.kms.model.PutKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.RevokeGrantRequest;
import software.amazon.awssdk.services.kms.model.ScheduleKeyDeletionRequest;
import software.amazon.awssdk.services.kms.model.SignRequest;
import software.amazon.awssdk.services.kms.model.SignResponse;
import software.amazon.awssdk.services.kms.model.SigningAlgorithmSpec;
import software.amazon.awssdk.services.kms.model.Tag;
import software.amazon.awssdk.services.kms.model.TagResourceRequest;
import software.amazon.awssdk.services.kms.model.VerifyRequest;
import software.amazon.awssdk.services.kms.model.VerifyResponse;
import software.amazon.awssdk.services.kms.paginators.ListAliasesIterable;
import software.amazon.awssdk.services.kms.paginators.ListGrantsIterable;
import software.amazon.awssdk.services.secretsmanager.model.ResourceExistsException;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;
import java.nio.ByteBuffer;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        final String usage = ""
            Usage: <granteePrincipal>

            Where:
                granteePrincipal - The principal (user, service account, or
                group) to whom the grant or permission is being given.
        """;
    }
}
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }
    String granteePrincipal = args[0];
    String policyName = "default";

    Scanner scanner = new Scanner(System.in);
    String keyDesc = "Created by the AWS KMS API";

    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("""
        Welcome to the AWS Key Management SDK Getting Started scenario.

        This program demonstrates how to interact with AWS Key Management using
the AWS SDK for Java (v2).
        The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
        and manage AWS KMS keys and control their use across a wide range of AWS
services and applications.
        KMS provides a centralized and unified approach to managing encryption
keys, making it easier to meet your
        data protection and regulatory compliance requirements.

        This Getting Started scenario creates two key types. A symmetric
encryption key is used to encrypt and decrypt data,
        and an asymmetric key used to digitally sign data.
        Let's get started...
        """);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("1. Create a symmetric KMS key\n");
    System.out.println("First, the program will creates a symmetric KMS key that
you can used to encrypt and decrypt data.");
    waitForInputToContinue(scanner);
    String targetKeyId = createKey(kmsClient, keyDesc);
    waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("""
    2. Enable a KMS key
```

By default, when the SDK creates an AWS key it is enabled. The next bit of code checks to determine if the key is enabled. If it is not enabled, the code enables it.

```
        """);
waitForInputToContinue(scanner);
boolean isEnabled = isKeyEnabled(kmsClient, targetKeyId);
if (!isEnabled)
    enableKey(kmsClient, targetKeyId);
waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("3. Encrypt data using the symmetric KMS key");
String plaintext = "Hello, AWS KMS!";
System.out.printf("""
```

One of the main uses of symmetric keys is to encrypt and decrypt data.

Next, the code encrypts the string '%s' with the SYMMETRIC_DEFAULT encryption algorithm.

```
        %n""", plaintext);
waitForInputToContinue(scanner);
SdkBytes ciphertext = encryptData(kmsClient, targetKeyId, plaintext);
waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("4. Create an alias");
System.out.println("""
```

Enter an alias name for the key. The name should be prefixed with 'alias/'.

For example, 'alias/myFirstKey'.

```
        """);
```

```
String aliasName = scanner.nextLine();
String fullAliasName = aliasName.isEmpty() ? "alias/dev-encryption-key" :
aliasName;
createCustomAlias(kmsClient, targetKeyId, fullAliasName);
waitForInputToContinue(scanner);
```

```

System.out.println(DASHES);
System.out.println("5. List all of your aliases");
waitForInputToContinue(scanner);
listAllAliases(kmsClient);
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("6. Enable automatic rotation of the KMS key");
System.out.println("

```

By default, when the SDK enables automatic rotation of a KMS key, KMS rotates the key material of the KMS key one year (approximately 365 days) from the enable date and every year thereafter.

```

""");
waitForInputToContinue(scanner);
enableKeyRotation(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("
    7. Create a grant

```

A grant is a policy instrument that allows Amazon Web Services principals to use KMS keys.

It also can allow them to view a KMS key (DescribeKey) and create and manage grants.

When authorizing access to a KMS key, grants are considered along with key policies and IAM policies.

```

""");

waitForInputToContinue(scanner);
String grantId = grantKey(kmsClient, targetKeyId, granteePrincipal);
System.out.println("The code granted principal with ARN [" +
granteePrincipal + "] ");
System.out.println("use of the symmetric key. The grant ID is [" + grantId +
"]");
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("8. List grants for the KMS key");
waitForInputToContinue(scanner);
displayGrantIds(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

```



```

System.out.println(DASHES);
System.out.println("9. Revoke the grant");
waitForInputToContinue(scanner);
revokeKeyGrant(kmsClient, targetKeyId, grantId);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("10. Decrypt the data\n");
System.out.println("

```

 Lets decrypt the data that was encrypted in an early step.
 The code uses the same key to decrypt the string that we encrypted
earlier in the program.

```

");
waitForInputToContinue(scanner);
String decryptText = decryptData(kmsClient, ciphertext, targetKeyId);
System.out.println("Decrypted text is: " + decryptText);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("11. Replace a key policy\n");
System.out.println("

```

 A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
 access to KMS keys. Every KMS key must have exactly one key policy. The
statements in the key policy
 determine who has permission to use the KMS key and how they can use
it.

 You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
 must have a key policy.

 By default, when you create a key by using the SDK, a policy is created
that
 gives the AWS account that owns the KMS key full access to the KMS key.

 Let's try to replace the automatically created policy with the following
policy.

```

    "Version": "2012-10-17",
    "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
    "Action": "kms:*",

```

```
        "Resource": "*"
    }]
    """);

    waitForInputToContinue(scanner);
    boolean polAdded = replacePolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("12. Get the key policy\n");
    System.out.println("The next bit of code that runs gets the key policy to
make sure it exists.");
    waitForInputToContinue(scanner);
    getKeyPolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("13. Create an asymmetric KMS key and sign your data\n");
    System.out.println("""
        Signing your data with an AWS key can provide several benefits that make
it an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage
the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
    """);
    waitForInputToContinue(scanner);
    signVerifyData(kmsClient);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("14. Tag your symmetric KMS Key\n");
    System.out.println("""
        By using tags, you can improve the overall management, security, and
governance of your
        KMS keys, making it easier to organize, track, and control access to
your encrypted data within
        your AWS environment
    """);
    waitForInputToContinue(scanner);
    tagKMSKey(kmsClient, targetKeyId);
    waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("15. Schedule the deletion of the KMS key\n");
System.out.println("""
    By default, KMS applies a waiting period of 30 days,
    but you can specify a waiting period of 7-30 days. When this operation
is successful,
    the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
    cryptographic operations. It remains in this state for the duration of
the waiting period.

    Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
    all data that was encrypted under the KMS key is unrecoverable.\s
""");
System.out.println("Would you like to delete the Key Management resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the AWS KMS resources.");
    waitForInputToContinue(scanner);
    deleteSpecificAlias(kmsClient, fullAliasName);
    disableKey(kmsClient, targetKeyId);
    deleteKey(kmsClient, targetKeyId);
} else {
    System.out.println("The Key Management resources will not be deleted");
}

System.out.println(DASHES);
System.out.println("This concludes the AWS Key Management SDK Getting
Started scenario");
System.out.println(DASHES);
}
public static void listAllAliases(KmsClient kmsClient) {
    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
        aliasesResponse.stream()
```

```
        .flatMap(r -> r.aliases().stream())
        .forEach(alias -> System.out
            .println("The alias name is: " + alias.aliasName()));

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void disableKey(KmsClient kmsClient, String keyId) {
    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void signVerifyData(KmsClient kmsClient) {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest request = CreateKeyRequest.builder()
        .keySpec(KeySpec.RSA_2048) // Specify key spec
        .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

    CreateKeyResponse response = kmsClient.createKey(request);
    String keyId2 = response.keyMetadata().keyId();
    System.out.println("Created KMS key with ID: " + keyId2);

    SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
    SignRequest signRequest = SignRequest.builder()
        .keyId(keyId2)
        .message(bytes)
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();
}
```

```
SignResponse signResponse = kmsClient.sign(signRequest);
byte[] signedBytes = signResponse.signature().asByteArray();

// Verify the digital signature.
VerifyRequest verifyRequest = VerifyRequest.builder()
    .keyId(keyId2)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset()))
    .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build());

VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}

public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
        System.out.println("The key has been tagged.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
```

```
        .policyName(policyName)
        .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}

public static boolean replacePolicy(KmsClient kmsClient, String keyId, String
policyName) {
    // Change the principle in the below JSON.
    String policy = ""
    {
        "Version": "2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::%s:root"},
            "Action": "kms:*",
            "Resource": "*"
        }]
    }
    """".formatted(accountId);

    try {
        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
            .build();
        kmsClient.putKeyPolicy(keyPolicyRequest);
        System.out.println("The key policy has been replaced.");
    } catch (LimitExceededException e) {
        System.out.println("Policy limit reached. Unable to create the
policy.");
        return false;
    } catch (AlreadyExistsException e) {
```

```
        System.out.println("Only one policy per key is supported. Unable to
create the policy.");
        return false;
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return true;
}

public static boolean doesKeyHavePolicy(KmsClient kmsClient, String keyId,
String policyName){
    ListKeyPoliciesRequest policiesRequest = ListKeyPoliciesRequest.builder()
        .keyId(keyId)
        .build();

    boolean hasPolicy = false;
    ListKeyPoliciesResponse response =
kmsClient.listKeyPolicies(policiesRequest);
    List<String>policyNames = response.policyNames();
    for (String pol : policyNames) {
        hasPolicy = true;
    }
    return hasPolicy;
}

public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        kmsClient.scheduleKeyDeletion(deletionRequest);
        System.out.println("The key will be deleted in 7 days.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
            System.out.println("The key is not enabled. Key state: " +
keyState);
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return false;
}

public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();
```



```
        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoke!");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsIterable response =
kmsClient.listGrantsPaginator(grantsRequest);
        response.stream()
            .flatMap(r -> r.grants().stream())
            .forEach(grant -> {
                System.out.println("The grant Id is : " + grant.grantId());
                List<GrantOperation> ops = grant.operations();
                for (GrantOperation op : ops) {
```

```
        System.out.println(op.name());
    }
});

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void enableKeyRotation(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKeyRotation(enableKeyRotationRequest);
    }
}
```

```
        System.out.println("Key rotation has been enabled for key with id [" +
keyId + "]);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
        try {
            CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
                .aliasName(aliasName)
                .targetKeyId(targetKeyId)
                .build();

            kmsClient.createAlias(aliasRequest);
            System.out.println(aliasName + " was successfully created.");

        } catch (ResourceExistsException e) {
            System.err.println("Alias already exists: " + e.getMessage());
            System.err.println("Moving on...");
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
            System.err.println("Moving on...");
        }
    }

    public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
        try {
            SdkBytes myBytes = SdkBytes.fromUtf8String(text);
            EncryptRequest encryptRequest = EncryptRequest.builder()
                .keyId(keyId)
                .plaintext(myBytes)
                .build();

            EncryptResponse response = kmsClient.encrypt(encryptRequest);
            String algorithm = response.encryptionAlgorithm().toString();
            System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

            // Get the encrypted data.
```

```
        SdkBytes encryptedData = response.ciphertextBlob();
        return encryptedData;

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}

public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Enable the KMS key.
public static void enableKey(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println("");
        System.out.println("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println("");
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}

private static String getAccountId(){
    try (StsClient stsClient = StsClient.create()){
        GetCallerIdentityResponse callerIdentity =
stsClient.getCallerIdentity();
        return callerIdentity.account();
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateKey](#)
 - [DescribeKey](#)
 - [DisableKey](#)
 - [EnableKey](#)
 - [GenerateDataKey](#)
 - [ListKeys](#)
 - [ScheduleKeyDeletion](#)
 - [Sign \(Signer\)](#)

Exemples Lambda utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide de Lambda.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Hello Lambda

Les exemples de code suivants montrent comment démarrer avec Lambda.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
package com.example.lambda;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListLambdaFunctions {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        listFunctions(awsLambda);
        awsLambda.close();
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListFunctions](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)
- [Exemples sans serveur](#)

Actions

CreateFunction

L'exemple de code suivant montre comment utiliser `CreateFunction`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.CreateFunctionRequest;
import software.amazon.awssdk.services.lambda.model.FunctionCode;
import software.amazon.awssdk.services.lambda.model.CreateFunctionResponse;
import software.amazon.awssdk.services.lambda.model.GetFunctionRequest;
import software.amazon.awssdk.services.lambda.model.GetFunctionResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.Runtime;
import software.amazon.awssdk.services.lambda.waiters.LambdaWaiter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

/**
 * This code example requires a ZIP or JAR that represents the code of the
 * Lambda function.
 * If you do not have a ZIP or JAR, please refer to the following document:
 *
 * https://github.com/aws-doc-sdk-examples/tree/master/javav2/usecases/
creating_workflows_stepfunctions
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```



```
*/

public class CreateFunction {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the ZIP or JAR where the code is located.
\s
                role - The role ARN that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
example.Handler::handleRequest). \s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        String role = args[2];
        String handler = args[3];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        createLambdaFunction(awsLambda, functionName, filePath, role, handler);
        awsLambda.close();
    }

    public static void createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
        String handler) {

        try {
```

```

LambdaWaiter waiter = awsLambda.waiter();
InputStream is = new FileInputStream(filePath);
SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

FunctionCode code = FunctionCode.builder()
    .zipFile(fileToUpload)
    .build();

CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
    .functionName(functionName)
    .description("Created by the Lambda Java API")
    .code(code)
    .handler(handler)
    .runtime(Runtime.JAVA8)
    .role(role)
    .build();

// Create a Lambda function using a waiter.
CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
    .functionName(functionName)
    .build();

WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("The function ARN is " +
functionResponse.functionArn());

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateFunction](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteFunction

L'exemple de code suivant montre comment utiliser `DeleteFunction`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFunction {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <functionName>\s

                Where:
                functionName - The name of the Lambda function.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_EAST_1;
        LambdaClient awsLambda = LambdaClient.builder()
                .region(region)
                .build();
```

```
        deleteLambdaFunction(awsLambda, functionName);
        awsLambda.close();
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteFunction](#) à la section Référence des AWS SDK for Java 2.x API.

Invoke

L'exemple de code suivant montre comment utiliser Invoke.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import org.json.JSONObject;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lambda.LambdaClient;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;

public class LambdaInvoke {

    /**
     * Function names appear as
     * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
     * you can retrieve the value by looking at the function in the AWS Console
     *
     * Also, set up your development environment, including your credentials.
     *
     * For information, see this documentation topic:
     *
     * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
     */

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName>\s

            Where:
                functionName - The name of the Lambda function\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        invokeFunction(awsLambda, functionName);
        awsLambda.close();
    }
}
```

```
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        // Setup an InvokeRequest.
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour en savoir plus sur l'API, consultez [Invoke](#) dans la Référence de l'API AWS SDK for Java 2.x .

Scénarios

Mise en route avec des fonctions

L'exemple de code suivant illustre comment :

- Créer un rôle IAM et une fonction Lambda, puis charger le code du gestionnaire.
- Invoquer la fonction avec un seul paramètre et obtenir des résultats.
- Mettre à jour le code de la fonction et configurer avec une variable d'environnement.

- Invoquer la fonction avec de nouveaux paramètres et obtenir des résultats. Afficher le journal d'exécution renvoyé.
- Répertorier les fonctions pour votre compte, puis nettoyer les ressources.

Pour plus d'informations, consultez [Créer une fonction Lambda à l'aide de la console](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/*
 * Lambda function names appear as:
 *
 * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 * To find this value, look at the function in the AWS Management Console.
 *
 * Before running this Java code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
public static void main(String[] args) throws InterruptedException {
    final String usage = ""

        Usage:
            <functionName> <filePath> <role> <handler> <bucketName> <key>\s

        Where:
            functionName - The name of the Lambda function.\s
            filePath - The path to the .zip or .jar where the code is
located.\s
            role - The AWS Identity and Access Management (IAM) service role
that has Lambda permissions.\s
            handler - The fully qualified method name (for example,
example.Handler::handleRequest).\s
            bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name that contains the .zip or .jar used to update the Lambda function's
code.\s
            key - The Amazon S3 key name that represents the .zip or .jar
(for example, LambdaHello-1.0-SNAPSHOT.jar).
        """;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    String role = args[2];
    String handler = args[3];
    String bucketName = args[4];
    String key = args[5];

    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Lambda example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
```



```
System.out.println("1. Create an AWS Lambda function.");
String funArn = createLambdaFunction(awsLambda, functionName, filePath,
role, handler);
System.out.println("The AWS Lambda ARN is " + funArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get the " + functionName + " AWS Lambda function.");
getFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List all AWS Lambda functions.");
listFunctions(awsLambda);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Invoke the Lambda function.");
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Update the Lambda function code and invoke it
again.");
updateFunctionCode(awsLambda, functionName, bucketName, key);
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Update a Lambda function's configuration value.");
updateFunctionConfiguration(awsLambda, functionName, handler);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the AWS Lambda function.");
LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS Lambda scenario completed successfully");
```

```
        System.out.println(DASHES);
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
        String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                .zipFile(fileToUpload)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA8)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

```
}

public static void getFunction(LambdaClient awsLambda, String functionName) {
    try {
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response = awsLambda.getFunction(functionRequest);
        System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listFunctions(LambdaClient awsLambda) {
    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();
        for (FunctionConfiguration config : list) {
            System.out.println("The function name is " + config.functionName());
        }

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
```

```
        .payload(payload)
        .build();

    res = awsLambda.invoke(request);
    String value = res.payload().asUtf8String();
    System.out.println(value);

} catch (LambdaException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
            .functionName(functionName)
            .publish(true)
            .s3Bucket(bucketName)
            .s3Key(key)
            .build();

        UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
        GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .build();

        WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
            .waitUntilFunctionUpdated(getFunctionConfigRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is " +
response.lastModified());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

}
```

```
public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
    try {
        UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .build();

        awsLambda.updateFunctionConfiguration(configurationRequest);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The " + functionName + " function was deleted");

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)

- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

Exemples sans serveur

Invoker une fonction Lambda à partir d'un déclencheur Kinesis

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements provenant d'un flux Kinesis. La fonction récupère la charge utile Kinesis, décode à partir de Base64 et enregistre le contenu de l'enregistrement.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Consommation d'un événement Kinesis avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
    }
}
```

```
    for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
        try {
            logger.log("Processed Event with EventId: "+record.getEventID());
            String data = new String(record.getKinesis().getData().array());
            logger.log("Data:"+ data);
            // TODO: Do interesting work based on the new data
        }
        catch (Exception ex) {
            logger.log("An error occurred:"+ex.getMessage());
            throw ex;
        }
    }
    logger.log("Successfully processed:"+event.getRecords().size()+" records");
    return null;
}
}
```

Invoquer une fonction lambda à partir d'un déclencheur Amazon S3

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par le téléchargement d'un objet dans un compartiment S3. La fonction extrait le nom du compartiment S3 et la clé de l'objet à partir du paramètre d'événement et appelle l'API Amazon S3 pour récupérer et consigner le type de contenu de l'objet.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement S3 avec Lambda en utilisant Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificationRecord;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```


Invocation d'une fonction lambda à partir d'un déclencheur Amazon SNS

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages provenant d'une rubrique SNS. La fonction extrait les messages du paramètre d'événement et consigne le contenu de chaque message.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement SNS avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
    }
}
```

```
    }
    return Boolean.TRUE;
}

public void processRecord(SNSRecord record) {
    try {
        String message = record.getSNS().getMessage();
        logger.log("message: " + message);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
}
```

Invoquer une fonction Lambda à partir d'un déclencheur Amazon SQS

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages provenant d'une file d'attente SQS. La fonction extrait les messages du paramètre d'événement et consigne le contenu de chaque message.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement SQS avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;
```

```
public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur Kinesis

L'exemple de code suivant montre comment implémenter une réponse par lots partielle pour les fonctions Lambda qui reçoivent des événements d'un flux Kinesis. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Signalement des échecs d'articles par lots Kinesis avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse(batchItemFailures);
    }
}
```

Signalement des défaillances d'éléments de lot pour les fonctions Lambda à l'aide d'un déclencheur DynamoDB

L'exemple de code suivant montre comment implémenter une réponse par lots partielle pour les fonctions Lambda qui reçoivent des événements d'un flux DynamoDB. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Signaler les défaillances d'éléments de lot DynamoDB avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<>();
```

```
String curRecordSequenceNumber = "";

for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
input.getRecords()) {
    try {
        //Process your record
        StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
        curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

    } catch (Exception e) {
        /* Since we are working with streams, we can return the failed item
immediately.
        Lambda will immediately begin to retry processing from this
failed item onwards. */
        batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
        return new StreamsEventResponse(batchItemFailures);
    }
}

return new StreamsEventResponse();
}
```

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur Amazon SQS

L'exemple de code suivant montre comment implémenter une réponse par lots partielle pour les fonctions Lambda qui reçoivent des événements d'une file d'attente SQS. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Signalement des échecs d'articles par lots SQS avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

MediaConvert exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with MediaConvert.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateJob

L'exemple de code suivant montre comment utiliser `CreateJob`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
```



```
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
    software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
```

```
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
    software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
    software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
    software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
```

```
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class CreateJob {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <mcRoleARN> <fileInput>\s

            Where:
                mcRoleARN - The MediaConvert Role ARN.\s
                fileInput - The URL of an Amazon S3 bucket
            where the input file is located.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String mcRoleARN = args[0];
        String fileInput = args[1];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        String id = createMediaJob(mc, mcRoleARN, fileInput);
        System.out.println("MediaConvert job created. Job Id = " + id);
        mc.close();
    }
}
```

```
    public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
String fileInput) {

        String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
1) + "javasdk/out/";
        String fileOutput = s3path + "index";
        String thumbsOutput = s3path + "thumbs/";
        String mp4Output = s3path + "mp4/";

        try {
            DescribeEndpointsResponse res = mc

.describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service
endpoint URL!");

                System.exit(1);
            }
            String endpointURL = res.endpoints().get(0).url();
            System.out.println("MediaConvert service URL: " +
endpointURL);

            System.out.println("MediaConvert role arn: " + mcRoleARN);
            System.out.println("MediaConvert input file: " + fileInput);
            System.out.println("MediaConvert output path: " + s3path);

            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            // output group Preset HLS low profile
            Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
750000, 7, 1920, 1080, 640);
            // output group Preset HLS media profile
            Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
            // output group Preset HLS high profole
            Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
3500000, 8, 1920, 1080, 1920);

            OutputGroup appleHLS = OutputGroup.builder().name("Apple
HLS").customName("Example")
```

```
.outputGroupSettings(OutputGroupSettings.builder()
    .type(OutputGroupType.HLS_GROUP_SETTINGS)
    .hlsGroupSettings(HlsGroupSettings.builder()
        .directoryStructure(
            HlsDirectoryStructure.SINGLE_DIRECTORY)
        .manifestDurationFormat(
            HlsManifestDurationFormat.INTEGER)
        .streamInfResolution(
            HlsStreamInfResolution.INCLUDE)
        .clientCache(HlsClientCache.ENABLED)
        .captionLanguageSetting(
            HlsCaptionLanguageSetting.OMIT)
        .manifestCompression(
            HlsManifestCompression.NONE)
        .codecSpecification(
            HlsCodecSpecification.RFC_4281)
        .outputSelection(
            HlsOutputSelection.MANIFESTS_AND_SEGMENTS)
        .programDateTime(HlsProgramDateTime.EXCLUDE)
        .programDateTimePeriod(600)
        .timedMetadataId3Frame(
            HlsTimedMetadataId3Frame.PRIV)
```

```

.timedMetadataId3Period(10)

.destination(fileOutput)

.segmentControl(HlsSegmentControl.SEGMENTED_FILES)

.minFinalSegmentLength((double) 0)

.segmentLength(4).minSegmentLength(0).build()
                                .build()
                                .outputs(hlsLow, hlsMedium,
hlsHigh).build());

        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(mp4Output).build()
                                .build()
                                .outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.MP4).build())

.videoDescription(VideoDescription.builder().width(1280)
                                .height(720)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

        VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

```

```
.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.H_264)

    .h264Settings(H264Settings

        .builder()

        .rateControlMode(

            H264RateControlMode.QVBR)

        .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

        .qualityTuningLevel(

            H264QualityTuningLevel.SINGLE_PASS)

        .qvbrSettings(

            H264QvbrSettings.builder()

                .qvbrQualityLevel(

                    8)

                .build())

        .codecLevel(H264CodecLevel.AUTO)

        .codecProfile(H264CodecProfile.MAIN)

        .maxBitrate(2400000)

        .framerateControl(

            H264FramerateControl.INITIALIZE_FROM_SOURCE)
```

```
.gopSize(2.0)

.gopSizeUnits(H264GopSizeUnits.SECONDS)

.numberBFramesBetweenReferenceFrames(
    2)

.gopClosedCadence(
    1)

.gopBReference(H264GopBReference.DISABLED)

.slowPal(H264SlowPal.DISABLED)

.syntax(H264Syntax.DEFAULT)

.numberReferenceFrames(
    3)

.dynamicSubGop(H264DynamicSubGop.STATIC)

.fieldEncoding(H264FieldEncoding.PAFF)

.sceneChangeDetect(
    H264SceneChangeDetect.ENABLED)

.minIInterval(0)

.telecine(H264Telecine.NONE)

.framerateConversionAlgorithm(
    H264FramerateConversionAlgorithm.DUPLICATE_DROP)

.entropyEncoding(
    H264EntropyEncoding.CABAC)

.slices(1)
```



```
.unregisteredSeiTimecode(  
    H264UnregisteredSeiTimecode.DISABLED)  
  
.repeatPps(H264RepeatPps.DISABLED)  
  
.adaptiveQuantization(  
    H264AdaptiveQuantization.HIGH)  
  
.spatialAdaptiveQuantization(  
    H264SpatialAdaptiveQuantization.ENABLED)  
  
.temporalAdaptiveQuantization(  
    H264TemporalAdaptiveQuantization.ENABLED)  
  
.flickerAdaptiveQuantization(  
    H264FlickerAdaptiveQuantization.DISABLED)  
  
.softness(0)  
  
.interlaceMode(H264InterlaceMode.PROGRESSIVE)  
  
.build()  
  
.build()                                     .build()  
  
.audioDescriptions(AudioDescription.builder())  
  
.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)  
  
.languageCodeControl(  
    AudioLanguageCodeControl.FOLLOW_INPUT)  
  
.codecSettings(AudioCodecSettings.builder())  
  
.codec(AudioCodec.AAC)
```

```

        .aacSettings(AacSettings
            .builder()
            .codecProfile(AacCodecProfile.LC)
            .rateControlMode(
                AacRateControlMode.CBR)
            .codingMode(AacCodingMode.CODING_MODE_2_0)
            .sampleRate(44100)
            .bitrate(160000)
            .rawFormat(AacRawFormat.NONE)
            .specification(AacSpecification.MPEG4)
            .audioDescriptionBroadcasterMix(
                AacAudioDescriptionBroadcasterMix.NORMAL)
            .build())
        .build()
        .build()
        .build()
        .build();
        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")
        .outputGroupSettings(OutputGroupSettings.builder()
        .type(OutputGroupType.FILE_GROUP_SETTINGS)
        .fileGroupSettings(FileGroupSettings.builder()
        .destination(thumbsOutput).build())
        .build()
        .outputs(Output.builder().extension("jpg")

```

```
.containerSettings(ContainerSettings.builder()
    .container(ContainerType.RAW).build())
    .videoDescription(VideoDescription.builder()
        .scalingBehavior(ScalingBehavior.DEFAULT)
        .sharpness(50).antiAlias(AntiAlias.ENABLED)
        .timecodeInsertion(
            VideoTimecodeInsertion.DISABLED)
        .colorMetadata(ColorMetadata.INSERT)
        .dropFrameTimecode(DropFrameTimecode.ENABLED)
        .codecSettings(VideoCodecSettings.builder()
            .codec(VideoCodec.FRAME_CAPTURE)
            .frameCaptureSettings(
                FrameCaptureSettings
                    .builder()
                        .framerateNumerator(
                            1)
                        .framerateDenominator(
                            1)
                        .maxCaptures(10000000)
                        .quality(80)
                    .build())
            .build())
    .build())
```

```

        .build()
        .build()
        .build();

        Map<String, AudioSelector> audioSelectors = new HashMap<>();
        audioSelectors.put("Audio Selector 1",
AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
        .offset(0).build());

        JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
        .audioSelectors(audioSelectors)
        .videoSelector(
VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)
        .rotate(InputRotate.DEGREE_0).build())
        .filterEnable(InputFilterEnable.AUTO).filterStrength(0)
        .deblockFilter(InputDeblockFilter.DISABLED)
        .denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)
        .timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
        .outputGroups(appleHLS, thumbs,
fileMp4).build());

        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
        .settings(jobSettings)
        .build();

        CreateJobResponse createJobResponse =
emc.createJob(createJobRequest);
        return createJobResponse.job().id();

    } catch (MediaConvertException e) {
        System.out.println(e.toString());
        System.exit(0);
    }
    return "";
}

```

```
private final static Output createOutput(String customName,
    String nameModifier,
    String segmentModifier,
    int qvbrMaxBitrate,
    int qvbrQualityLevel,
    int originWidth,
    int originHeight,
    int targetWidth) {

    int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
    - (Math.round(originHeight * targetWidth /
originWidth) % 4);
    Output output = null;
    try {
        output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
        .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,
```

```
489, 490, 491, 492)
                                                                    .build())
                                                                    .build())
                                                                    .videoDescription(
VideoDescription.builder().width(targetWidth)
.height(targetHeight)
.scalingBehavior(ScalingBehavior.DEFAULT)
.sharpness(50).antiAlias(AntiAlias.ENABLED)
.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)
.colorMetadata(ColorMetadata.INSERT)
.respondToAfd(RespondToAfd.NONE)
.afdSignaling(AfdSignaling.NONE)
.dropFrameTimecode(DropFrameTimecode.ENABLED)
.codecSettings(VideoCodecSettings.builder()
    .codec(VideoCodec.H_264)
    .h264Settings(H264Settings
        .builder()
        .rateControlMode(
            H264RateControlMode.QVBR)
        .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
        .qualityTuningLevel(
            H264QualityTuningLevel.SINGLE_PASS)
```

```
.qvbrSettings(H264QvbrSettings
    .builder()
    .qvbrQualityLevel(
        qvbrQualityLevel)
    .build())
.codecLevel(H264CodecLevel.AUTO)
.codecProfile((targetHeight > 720
    && targetWidth > 1280)
    ? H264CodecProfile.HIGH
    : H264CodecProfile.MAIN)
.maxBitrate(qvbrMaxBitrate)
.framerateControl(
    H264FramerateControl.INITIALIZE_FROM_SOURCE)
.gopSize(2.0)
.gopSizeUnits(H264GopSizeUnits.SECONDS)
.numberBFramesBetweenReferenceFrames(
    2)
.gopClosedCadence(
    1)
.gopBReference(H264GopBReference.DISABLED)
.slowPal(H264SlowPal.DISABLED)
.syntax(H264Syntax.DEFAULT)
```

```
.numberReferenceFrames(  
    3)  
  
.dynamicSubGop(H264DynamicSubGop.STATIC)  
  
.fieldEncoding(H264FieldEncoding.PAFF)  
  
.sceneChangeDetect(  
    H264SceneChangeDetect.ENABLED)  
  
.minIInterval(0)  
  
.telecine(H264Telecine.NONE)  
  
.framerateConversionAlgorithm(  
    H264FramerateConversionAlgorithm.DUPLICATE_DROP)  
  
.entropyEncoding(  
    H264EntropyEncoding.CABAC)  
  
.slices(1)  
  
.unregisteredSeiTimecode(  
    H264UnregisteredSeiTimecode.DISABLED)  
  
.repeatPps(H264RepeatPps.DISABLED)  
  
.adaptiveQuantization(  
    H264AdaptiveQuantization.HIGH)  
  
.spatialAdaptiveQuantization(  
    H264SpatialAdaptiveQuantization.ENABLED)  
  
.temporalAdaptiveQuantization(  
    H264TemporalAdaptiveQuantization.ENABLED)
```



```
        .flickerAdaptiveQuantization(
            H264FlickerAdaptiveQuantization.DISABLED)

        .softness(0)

        .interlaceMode(H264InterlaceMode.PROGRESSIVE)

        .build()

    .build()

                                                                    .build()

    .audioDescriptions(AudioDescription.builder()

    .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

    .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

    .codecSettings(AudioCodecSettings.builder()

    .codec(AudioCodec.AAC).aacSettings(AacSettings

        .builder()

        .codecProfile(AacCodecProfile.LC)

        .rateControlMode(

            AacRateControlMode.CBR)

        .codingMode(AacCodingMode.CODING_MODE_2_0)

        .sampleRate(44100)

        .bitrate(96000)

        .rawFormat(AacRawFormat.NONE)

        .specification(AacSpecification.MPEG4)

        .audioDescriptionBroadcasterMix(
```

```

        AACAudioDescriptionBroadcasterMix.NORMAL)

        .build()

        .build()

        .build()

        .build();
    } catch (MediaConvertException e) {
        e.printStackTrace();
        System.exit(0);
    }
    return output;
}
}
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateJob](#) à la section Référence des AWS SDK for Java 2.x API.

GetJob

L'exemple de code suivant montre comment utiliser `GetJob`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**

```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
            " <jobId> \n\n" +
            "Where:\n" +
            " jobId - The job id value.\n\n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
            DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
                .maxResults(20)
                .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint
URL!");
                System.exit(1);
            }
            String endpointURL = res.endpoints().get(0).url();
        }
    }
}
```

```
MediaConvertClient emc = MediaConvertClient.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create(endpointURL))
    .build();

GetJobRequest jobRequest = GetJobRequest.builder()
    .id(jobId)
    .build();

GetJobResponse response = emc.getJob(jobRequest);
System.out.println("The ARN of the job is " + response.job().arn());

} catch (MediaConvertException e) {
    System.out.println(e.toString());
    System.exit(0);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetJob](#) à la section Référence des AWS SDK for Java 2.x API.

ListJobs

L'exemple de code suivant montre comment utiliser `ListJobs`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
```

```
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
                .maxResults(20)
                .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint
URL!");
                System.exit(1);
            }

            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            ListJobsRequest jobsRequest = ListJobsRequest.builder()
```

```
        .maxResults(10)
        .status("COMPLETE")
        .build();

    ListJobsResponse jobsResponse = emc.listJobs(jobsRequest);
    List<Job> jobs = jobsResponse.jobs();
    for (Job job : jobs) {
        System.out.println("The JOB ARN is : " + job.arn());
    }

} catch (MediaConvertException e) {
    System.out.println(e.toString());
    System.exit(0);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListJobs](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples de Migration Hub utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with Migration Hub.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

DeleteProgressUpdateStream

L'exemple de code suivant montre comment utiliser `DeleteProgressUpdateStream`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <progressStream>\s

            Where:
                progressStream - the name of a progress stream to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String progressStream = args[0];
    Region region = Region.US_WEST_2;
    MigrationHubClient migrationClient = MigrationHubClient.builder()
        .region(region)
        .build();

    deleteStream(migrationClient, progressStream);
    migrationClient.close();
}

public static void deleteStream(MigrationHubClient migrationClient, String
streamName) {
    try {
        DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
DeleteProgressUpdateStreamRequest
            .builder()
            .progressUpdateStreamName(streamName)
            .build();

        migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
        System.out.println(streamName + " is deleted");

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteProgressUpdateStream](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeApplicationState

L'exemple de code suivant montre comment utiliser `DescribeApplicationState`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                DescribeAppState <appId>\s

                Where:
                appId - the application id value.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        Region region = Region.US_WEST_2;
```

```
MigrationHubClient migrationClient = MigrationHubClient.builder()
    .region(region)
    .build();

describeApplicationState(migrationClient, appId);
migrationClient.close();
}

public static void describeApplicationState(MigrationHubClient migrationClient,
String appId) {
    try {
        DescribeApplicationStateRequest applicationStateRequest =
DescribeApplicationStateRequest.builder()
            .applicationId(appId)
            .build();

        DescribeApplicationStateResponse applicationStateResponse =
migrationClient
            .describeApplicationState(applicationStateRequest);
        System.out.println("The application status is " +
applicationStateResponse.applicationStatusAsString());

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeApplicationState](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeMigrationTask

L'exemple de code suivant montre comment utiliser `DescribeMigrationTask`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                DescribeMigrationTask <migrationTask> <progressStream>\s

            Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s

            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        describeMigTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void describeMigTask(MigrationHubClient migrationClient, String
migrationTask,
        String progressStream) {
        try {
            DescribeMigrationTaskRequest migrationTaskRequestRequest =
DescribeMigrationTaskRequest.builder()
                .progressUpdateStream(progressStream)
                .migrationTaskName(migrationTask)
                .build();

            DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
                .describeMigrationTask(migrationTaskRequestRequest);
            System.out.println("The name is " +
migrationTaskResponse.migrationTask().migrationTaskName());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeMigrationTask](#) à la section Référence des AWS SDK for Java 2.x API.

ImportMigrationTask

L'exemple de code suivant montre comment utiliser `ImportMigrationTask`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <migrationTask> <progressStream>\s

                Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
```

```

String progressStream = args[1];
Region region = Region.US_WEST_2;
MigrationHubClient migrationClient = MigrationHubClient.builder()
    .region(region)
    .build();

importMigrTask(migrationClient, migrationTask, progressStream);
migrationClient.close();
}

public static void importMigrTask(MigrationHubClient migrationClient, String
migrationTask, String progressStream) {
    try {
        CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
CreateProgressUpdateStreamRequest.builder()
            .progressUpdateStreamName(progressStream)
            .dryRun(false)
            .build();

        migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
        ImportMigrationTaskRequest migrationTaskRequest =
ImportMigrationTaskRequest.builder()
            .migrationTaskName(migrationTask)
            .progressUpdateStream(progressStream)
            .dryRun(false)
            .build();

        migrationClient.importMigrationTask(migrationTaskRequest);

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Pour plus de détails sur l'API, reportez-vous [ImportMigrationTask](#) à la section Référence des AWS SDK for Java 2.x API.

ListApplications

L'exemple de code suivant montre comment utiliser `ListApplications`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
import
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listApps(migrationClient);
        migrationClient.close();
    }

    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
                ListApplicationStatesRequest.builder()
                    .maxResults(10)
```

```
        .build();

        ListApplicationStatesResponse response =
migrationClient.listApplicationStates(applicationStatesRequest);
        List<ApplicationState> apps = response.applicationStateList();
        for (ApplicationState appState : apps) {
            System.out.println("App Id is " + appState.applicationId());
            System.out.println("The status is " +
appState.applicationStatus().toString());
        }

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListApplications](#) à la section Référence des AWS SDK for Java 2.x API.

ListCreatedArtifacts

L'exemple de code suivant montre comment utiliser `ListCreatedArtifacts`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
```



```
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
                ListCreatedArtifactsRequest.builder()
                    .maxResults(10)
                    .migrationTaskName("SampleApp5")
                    .progressUpdateStream("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
                migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("APp Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
            }
        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [ListCreatedArtifacts](#) à la section Référence des AWS SDK for Java 2.x API.

ListMigrationTasks

L'exemple de code suivant montre comment utiliser `ListMigrationTasks`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();
    }
}
```

```
        listMigrTasks(migrationClient);
        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
ListMigrationTasksRequest.builder()
                .maxResults(10)
                .build();

            ListMigrationTasksResponse response =
migrationClient.listMigrationTasks(listMigrationTasksRequest);
            List<MigrationTaskSummary> migrationList =
response.migrationTaskSummaryList();
            for (MigrationTaskSummary migration : migrationList) {
                System.out.println("Migration task name is " +
migration.migrationTaskName());
                System.out.println("The Progress update stream is " +
migration.progressUpdateStream());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListMigrationTasks](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples Amazon Personalize à l'aide du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Personalize.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateBatchInferenceJob

L'exemple de code suivant montre comment utiliser `CreateBatchInferenceJob`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                String solutionVersionArn,
                String jobName,
                String s3InputDataSourcePath,
                String s3DataDestinationPath,
                String roleArn,
                String explorationWeight,
                String explorationItemAgeCutOff) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String batchInferenceJobArn;

    try {
```

```
// Set up data input and output parameters.
S3DataConfig inputSource = S3DataConfig.builder()
    .path(s3InputDataSourcePath)
    .build();

S3DataConfig outputDestination = S3DataConfig.builder()
    .path(s3DataDestinationPath)
    .build();

BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
    .s3DataSource(inputSource)
    .build();

BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
    .s3DataDestination(outputDestination)
    .build();

// Optional code to build the User-Personalization specific
item exploration
// config.
HashMap<String, String> explorationConfig = new HashMap<>();

explorationConfig.put("explorationWeight",
explorationWeight);
explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
    .itemExplorationConfig(explorationConfig)
    .build();

// End optional User-Personalization recipe specific code.

CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
    .builder()
    .solutionVersionArn(solutionVersionArn)
    .jobInput(jobInput)
    .jobOutput(jobOutputLocation)
    .jobName(jobName)
```

```

        .roleArn(roleArn)
        .batchInferenceJobConfig(jobConfig) //
Optional
        .build();

        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
        .batchInferenceJobArn();

        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
        .builder()
        .batchInferenceJobArn(batchInferenceJobArn)
        .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
        while (Instant.now().getEpochSecond() < maxTime) {

                BatchInferenceJob batchInferenceJob =
personalizeClient

                .describeBatchInferenceJob(describeBatchInferenceJobRequest)
                        .batchInferenceJob();

                status = batchInferenceJob.status();
                System.out.println("Batch inference job status: " +
status);

                if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {

                        break;
                }
                try {
                        Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                }
        }
        return batchInferenceJobArn;

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";

```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBatchInferenceJob](#) à la section Référence des AWS SDK for Java 2.x API.

CreateCampaign

L'exemple de code suivant montre comment utiliser `CreateCampaign`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createPersonalCampaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
String name) {

    try {
        CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
            .minProvisionedTPS(1)
            .solutionVersionArn(solutionVersionArn)
            .name(name)
            .build();

        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCampaign](#) à la section Référence des AWS SDK for Java 2.x API.

CreateDataset

L'exemple de code suivant montre comment utiliser `CreateDataset`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createDataset(PersonalizeClient personalizeClient,
    String datasetName,
    String datasetGroupArn,
    String datasetType,
    String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
            .name(datasetName)
            .datasetGroupArn(datasetGroupArn)
            .datasetType(datasetType)
            .schemaArn(schemaArn)
            .build();

        String datasetArn = personalizeClient.createDataset(request)
            .datasetArn();
        System.out.println("Dataset " + datasetName + " created.");
        return datasetArn;
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```


- Pour plus de détails sur l'API, reportez-vous [CreateDataset](#) à la section Référence des AWS SDK for Java 2.x API.

CreateDatasetExportJob

L'exemple de code suivant montre comment utiliser `CreateDatasetExportJob`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,
    String jobName,
    String datasetArn,
    IngestionMode ingestionMode,
    String roleArn,
    String s3BucketPath,
    String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {

        S3DataConfig exportS3DataConfig =
        S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
        DatasetExportJobOutput jobOutput =
        DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
            .build();

        CreateDatasetExportJobRequest createRequest =
        CreateDatasetExportJobRequest.builder()
            .jobName(jobName)
            .datasetArn(datasetArn)
            .ingestionMode(ingestionMode)
            .jobOutput(jobOutput)
            .roleArn(roleArn)
            .build();
```

```

        String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

        DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
            .datasetExportJobArn(datasetExportJobArn)
            .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetExportJob datasetExportJob = personalizeClient
                .describeDatasetExportJob(describeDatasetExportJobRequest)
                .datasetExportJob();

            status = datasetExportJob.status();
            System.out.println("Export job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                return status;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateDatasetExportJob](#) à la section Référence des AWS SDK for Java 2.x API.

CreateDatasetGroup

L'exemple de code suivant montre comment utiliser CreateDatasetGroup.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

Créez un groupe de jeux de données de domaine.

```
public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
    String datasetGroupName,
    String domain) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .domain(domain)
            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
    }  
    return "";  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDatasetGroup](#) à la section Référence des AWS SDK for Java 2.x API.

CreateDatasetImportJob

L'exemple de code suivant montre comment utiliser `CreateDatasetImportJob`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient  
personalizeClient,  
    String jobName,  
    String datasetArn,  
    String s3BucketPath,  
    String roleArn) {  
  
    long waitInMilliseconds = 60 * 1000;  
    String status;  
    String datasetImportJobArn;  
  
    try {  
        DataSource importDataSource = DataSource.builder()  
            .dataLocation(s3BucketPath)  
            .build();  
  
        CreateDatasetImportJobRequest createDatasetImportJobRequest =  
        CreateDatasetImportJobRequest.builder()  
            .datasetArn(datasetArn)  
            .dataSource(importDataSource)  
            .jobName(jobName)
```

```
        .roleArn(roleArn)
        .build();

    datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
        .datasetImportJobArn();
    DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
        .datasetImportJobArn(datasetImportJobArn)
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetImportJob datasetImportJob = personalizeClient
            .describeDatasetImportJob(describeDatasetImportJobRequest)
            .datasetImportJob();

        status = datasetImportJob.status();
        System.out.println("Dataset import job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return datasetImportJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDatasetImportJob](#) à la section Référence des AWS SDK for Java 2.x API.

CreateEventTracker

L'exemple de code suivant montre comment utiliser `CreateEventTracker`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
    String datasetGroupArn) {

    String eventTrackerId = "";
    String eventTrackerArn;
    long maxTime = 3 * 60 * 60; // 3 hours
    long waitInMilliseconds = 20 * 1000; // 20 seconds
    String status;

    try {

        CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
            .name(eventTrackerName)
            .datasetGroupArn(datasetGroupArn)
            .build();

        CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
            .createEventTracker(createEventTrackerRequest);

        eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
        eventTrackerId = createEventTrackerResponse.trackingId();
        System.out.println("Event tracker ARN: " + eventTrackerArn);
        System.out.println("Event tracker ID: " + eventTrackerId);

        maxTime = Instant.now().getEpochSecond() + maxTime;

        DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
```

```
        .eventTrackerArn(eventTrackerArn)
        .build();

    while (Instant.now().getEpochSecond() < maxTime) {

        status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
        System.out.println("EventTracker status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return eventTrackerId;
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return eventTrackerId;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateEventTracker](#) à la section Référence des AWS SDK for Java 2.x API.

CreateFilter

L'exemple de code suivant montre comment utiliser `CreateFilter`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createFilter(PersonalizeClient personalizeClient,
    String filterName,
    String datasetGroupArn,
    String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateFilter](#) à la section Référence des AWS SDK for Java 2.x API.

CreateRecommender

L'exemple de code suivant montre comment utiliser `CreateRecommender`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createRecommender(PersonalizeClient personalizeClient,
    String name,
    String datasetGroupArn,
    String recipeArn) {

    long maxTime = 0;
```



```
long waitInMilliseconds = 30 * 1000; // 30 seconds
String recommenderStatus = "";

try {
    CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
        .datasetGroupArn(datasetGroupArn)
        .name(name)
        .recipeArn(recipeArn)
        .build();

    CreateRecommenderResponse recommenderResponse = personalizeClient
        .createRecommender(createRecommenderRequest);
    String recommenderArn = recommenderResponse.recommenderArn();
    System.out.println("The recommender ARN is " + recommenderArn);

    DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
        .recommenderArn(recommenderArn)
        .build();

    maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
            .status();
        System.out.println("Recommender status: " + recommenderStatus);

        if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return recommenderArn;

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}
```

```
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateRecommender](#) à la section Référence des AWS SDK for Java 2.x API.

CreateSchema

L'exemple de code suivant montre comment utiliser `CreateSchema`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);
    }
}
```

```
        return schemaArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Créez un schéma avec un domaine.

```
public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
    String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .domain(domain)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateSchema](#) à la section Référence des AWS SDK for Java 2.x API.

CreateSolution

L'exemple de code suivant montre comment utiliser `CreateSolution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
        String datasetGroupArn,
        String solutionName,
        String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
        return solutionResponse.solutionArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateSolution](#) à la section Référence des AWS SDK for Java 2.x API.

CreateSolutionVersion

L'exemple de code suivant montre comment utiliser `CreateSolutionVersion`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);
```

```
        if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }

    if (solutionStatus.equals("ACTIVE")) {

        CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
            .createSolutionVersion(createSolutionVersionRequest);
        solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

        System.out.println("Solution version ARN: " + solutionVersionArn);

        DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
            .solutionVersionArn(solutionVersionArn)
            .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                .solutionVersion().status();
            System.out.println("Solution version status: " +
solutionVersionStatus);

            if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                break;
            }
        }
        try {
```

```
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return solutionVersionArn;
}
} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateSolutionVersion](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteCampaign

L'exemple de code suivant montre comment utiliser `DeleteCampaign`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        personalizeClient.deleteCampaign(campaignRequest);

    } catch (PersonalizeException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCampaign](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteEventTracker

L'exemple de code suivant montre comment utiliser `DeleteEventTracker`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```


- Pour plus de détails sur l'API, reportez-vous [DeleteEventTracker](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteSolution

L'exemple de code suivant montre comment utiliser `DeleteSolution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        personalizeClient.deleteSolution(solutionRequest);
        System.out.println("Done");


    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSolution](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeCampaign

L'exemple de code suivant montre comment utiliser `DescribeCampaign`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign myCampaign = campaignResponse.campaign();
        System.out.println("The Campaign name is " + myCampaign.name());
        System.out.println("The Campaign status is " + myCampaign.status());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCampaign](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeRecipe

L'exemple de code suivant montre comment utiliser `DescribeRecipe`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

    try {
        DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
            .recipeArn(recipeArn)
            .build();

        DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
        System.out.println("The recipe name is " +
recipeResponse.recipe().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeRecipe](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeSolution

L'exemple de code suivant montre comment utiliser `DescribeSolution`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
        System.out.println("The Solution name is " +
response.solution().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSolution](#) à la section Référence des AWS SDK for Java 2.x API.

ListCampaigns

L'exemple de code suivant montre comment utiliser `ListCampaigns`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

    try {
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
            .maxResults(10)
            .solutionArn(solutionArn)
            .build();

        ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
        List<CampaignSummary> campaigns = response.campaigns();
        for (CampaignSummary campaign : campaigns) {
            System.out.println("Campaign name is : " + campaign.name());
            System.out.println("Campaign ARN is : " + campaign.campaignArn());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListCampaigns](#) à la section Référence des AWS SDK for Java 2.x API.

ListDatasetGroups

L'exemple de code suivant montre comment utiliser ListDatasetGroups.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listDSGroups(PersonalizeClient personalizeClient) {

    try {
        ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
        .maxResults(15)
        .build();

        ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
        List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
        for (DatasetGroupSummary group : groups) {
            System.out.println("The DataSet name is : " + group.name());
            System.out.println("The DataSet ARN is : " +
group.datasetGroupArn());
        }


    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDatasetGroups](#) à la section Référence des AWS SDK for Java 2.x API.

ListRecipes

L'exemple de code suivant montre comment utiliser `ListRecipes`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
public static void listAllRecipes(PersonalizeClient personalizeClient) {  
  
    try {  
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()  
            .maxResults(15)  
            .build();  
  
        ListRecipesResponse response =  
personalizeClient.listRecipes(recipesRequest);  
        List<RecipeSummary> recipes = response.recipes();  
        for (RecipeSummary recipe : recipes) {  
            System.out.println("The recipe ARN is: " + recipe.recipeArn());  
            System.out.println("The recipe name is: " + recipe.name());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [ListRecipes](#) à la section Référence des AWS SDK for Java 2.x API.

ListSolutions

L'exemple de code suivant montre comment utiliser `ListSolutions`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {

    try {
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
            .maxResults(10)
            .datasetGroupArn(datasetGroupArn)
            .build();

        ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
        List<SolutionSummary> solutions = response.solutions();
        for (SolutionSummary solution : solutions) {
            System.out.println("The solution ARN is: " +
solution.solutionArn());
            System.out.println("The solution name is: " + solution.name());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListSolutions](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateCampaign

L'exemple de code suivant montre comment utiliser UpdateCampaign.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String updateCampaign(PersonalizeClient personalizeClient,
    String campaignArn,
    String solutionVersionArn,
    Integer minProvisionedTPS) {

    try {
        // build the updateCampaignRequest
        UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
            .campaignArn(campaignArn)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .build();

        // update the campaign
        personalizeClient.updateCampaign(updateCampaignRequest);

        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign updatedCampaign = campaignResponse.campaign();

        System.out.println("The Campaign status is " +
updatedCampaign.status());
        return updatedCampaign.status();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }
```

- Pour plus de détails sur l'API, reportez-vous [UpdateCampaign](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'événements Amazon Personalize à l'aide du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Personalize Events.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

PutEvents

L'exemple de code suivant montre comment utiliserPutEvents.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String item1Id,
    String item1PropertyName,
    String item1PropertyValue,
    String item2Id,
    String item2PropertyName,
    String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{ \"%1$s\": \"%2$s
\}"),
                item1PropertyName,
                item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{ \"%1$s\": \"%2$s
\}"),
                item2PropertyName,
                item2PropertyValue))
            .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
            .datasetArn(datasetArn)
            .items(items)
            .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;
    }
}
```

```
    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

- Pour plus de détails sur l'API, reportez-vous [PutEvents](#) à la section Référence des AWS SDK for Java 2.x API.

PutUsers

L'exemple de code suivant montre comment utiliser PutUsers.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String user1Id,
    String user1PropertyName,
    String user1PropertyValue,
    String user2Id,
    String user2PropertyName,
    String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

    try {
        User user1 = User.builder()
            .userId(user1Id)
            .properties(String.format("{ \"%1$s\": \"%2$s
\"]]\",
                                user1PropertyName,
                                user1PropertyValue))
```

```

        .build());

        users.add(user1);

        User user2 = User.builder()
            .userId(user2Id)
            .properties(String.format("{\"%1$s\": \"%2$s
\"]]\",
                                user2PropertyName,
                                user2PropertyValue))
            .build();

        users.add(user2);

        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
            .datasetArn(datasetArn)
            .users(users)
            .build();

        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}

```

- Pour plus de détails sur l'API, reportez-vous [PutUsers](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'Amazon Personalize Runtime à l'aide du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Personalize Runtime.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

GetPersonalizedRanking

L'exemple de code suivant montre comment utiliser `GetPersonalizedRanking`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
        String campaignArn,
        String userId,
        ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
        GetPersonalizedRankingRequest.builder()
            .campaignArn(campaignArn)
            .userId(userId)
            .inputList(items)
            .build();
```

```
        GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient
            .getPersonalizedRanking(rankingRecommendationsRequest);
        List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
            System.out.println("-----");
            rank++;
        }
        return rankedItems;
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetPersonalizedRanking](#) à la section Référence des AWS SDK for Java 2.x API.

GetRecommendations

L'exemple de code suivant montre comment utiliser `GetRecommendations`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez une liste des articles recommandés.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {
```

```
    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();
        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Obtenez une liste d'éléments recommandés à partir d'un outil de recommandation créé dans un groupe de jeux de données de domaine.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
    String userId) {

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .recommenderArn(recommenderArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();
```



```

        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Utilisez un filtre lorsque vous demandez des recommandations.

```

public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
    String campaignArn,
    String userId,
    String filterArn,
    String parameter1Name,
    String parameter1Value1,
    String parameter1Value2,
    String parameter2Name,
    String parameter2Value) {

    try {

        Map<String, String> filterValues = new HashMap<>();

        filterValues.put(parameter1Name, String.format("%1$s\", \"%2$s\"",
            parameter1Value1, parameter1Value2));
        filterValues.put(parameter2Name, String.format("%1$s\"",
            parameter2Value));

        GetRecommendationsRequest recommendationsRequest =
        GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .filterArn(filterArn)
            .filterValues(filterValues)
            .build();
    }
}

```

```
        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetRecommendations](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'Amazon Pinpoint utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Pinpoint.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateApp

L'exemple de code suivant montre comment utiliser `CreateApp`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = ""

            Usage: <appName>

            Where:
            appName - The name of the application to create.

            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```

        System.exit(1);
    }
    String appName = args[0];
    System.out.println("Creating an application with name: " + appName);

    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String appID = createApplication(pinpoint, appName);
    System.out.println("App ID is: " + appID);
    pinpoint.close();
}

public static String createApplication(PinpointClient pinpoint, String appName)
{
    try {
        CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
            .name(appName)
            .build();

        CreateAppRequest request = CreateAppRequest.builder()
            .createApplicationRequest(appRequest)
            .build();

        CreateAppResponse result = pinpoint.createApp(request);
        return result.applicationResponse().id();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateApp](#) à la section Référence des AWS SDK for Java 2.x API.

CreateCampaign

L'exemple de code suivant montre comment utiliser `CreateCampaign`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créer une campagne.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCampaign {
    public static void main(String[] args) {

        final String usage = ""

            Usage:  <appId> <segmentId>

            Where:
                appId - The ID of the application to create the campaign in.
                segmentId - The ID of the segment to create the campaign from.
            """;

        if (args.length != 2) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String appId = args[0];
    String segmentId = args[1];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPinCampaign(pinpoint, appId, segmentId);
    pinpoint.close();
}

public static void createPinCampaign(PinpointClient pinpoint, String appId,
String segmentId) {
    CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
    System.out.println("Campaign " + result.name() + " created.");
    System.out.println(result.description());
}

public static CampaignResponse createCampaign(PinpointClient client, String
appId, String segmentID) {

    try {
        Schedule schedule = Schedule.builder()
            .startTime("IMMEDIATE")
            .build();

        Message defaultMessage = Message.builder()
            .action(Action.OPEN_APP)
            .body("My message body.")
            .title("My message title.")
            .build();

        MessageConfiguration messageConfiguration =
MessageConfiguration.builder()
            .defaultMessage(defaultMessage)
            .build();

        WriteCampaignRequest request = WriteCampaignRequest.builder()
            .description("My description")
            .schedule(schedule)
            .name("MyCampaign")
```

```
        .segmentId(segmentID)
        .messageConfiguration(messageConfiguration)
        .build();

    CreateCampaignResponse result =
client.createCampaign(CreateCampaignRequest.builder()
        .applicationId(appID)
        .writeCampaignRequest(request).build());

    System.out.println("Campaign ID: " + result.campaignResponse().id());
    return result.campaignResponse();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return null;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCampaign](#) à la section Référence des AWS SDK for Java 2.x API.

CreateExportJob

L'exemple de code suivant montre comment utiliser `CreateExportJob`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exporter un point de terminaison.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
```

```
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = ""
```

This program performs the following steps:

1. Exports the endpoints to an Amazon S3 bucket.
 2. Downloads the exported endpoints files from Amazon S3.
 3. Parses the endpoints files to obtain the endpoint IDs and prints them.

Usage: ExportEndpoints <applicationId> <s3BucketName>
 <iamExportRoleArn> <path>

Where:

applicationId - The ID of the Amazon Pinpoint application that has the endpoint.

s3BucketName - The name of the Amazon S3 bucket to export the JSON file to.\s

iamExportRoleArn - The ARN of an IAM role that grants Amazon Pinpoint write permissions to the S3 bucket. path - The path where the files downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).

```

    """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String applicationId = args[0];
    String s3BucketName = args[1];
    String iamExportRoleArn = args[2];
    String path = args[3];
    System.out.println("Deleting an application with ID: " + applicationId);

    Region region = Region.US_EAST_1;
    PinpointClient pinpoint = PinpointClient.builder()
        .region(region)
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
    pinpoint.close();
    s3Client.close();
}

public static void exportAllEndpoints(PinpointClient pinpoint,

```

```

        S3Client s3Client,
        String applicationId,
        String s3BucketName,
        String path,
        String iamExportRoleArn) {

    try {
        List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
s3BucketName, iamExportRoleArn,
            applicationId);
        List<String> endpointFileKeys = objectKeys.stream().filter(o ->
o.endsWith(".gz"))
            .collect(Collectors.toList());
        downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
s3Client, String s3BucketName,
        String iamExportRoleArn, String applicationId) {

    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
    String endpointsKeyPrefix = "exports/" + applicationId + "_" +
dateFormat.format(new Date());
    String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
"/";
    List<String> objectKeys = new ArrayList<>();
    String key;

    try {
        // Defines the export job that Amazon Pinpoint runs.
        ExportJobRequest jobRequest = ExportJobRequest.builder()
            .roleArn(iamExportRoleArn)
            .s3UrlPrefix(s3UrlPrefix)
            .build();

        CreateExportJobRequest exportJobRequest =
CreateExportJobRequest.builder()
            .applicationId(applicationId)

```

```
        .exportJobRequest(jobRequest)
        .build();

    System.out.format("Exporting endpoints from Amazon Pinpoint application
%s to Amazon S3 " +
        "bucket %s . . .\n", applicationId, s3BucketName);

    CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
    String jobId = exportResult.exportJobResponse().id();
    System.out.println(jobId);
    printExportJobStatus(pinpoint, applicationId, jobId);

    ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
        .bucket(s3BucketName)
        .prefix(endpointsKeyPrefix)
        .build();

    // Create a list of object keys.
    ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
    List<S3Object> objects = v2Response.contents();
    for (S3Object object : objects) {
        key = object.key();
        objectKeys.add(key);
    }

    return objectKeys;

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

private static void printExportJobStatus(PinpointClient pinpointClient,
    String applicationId,
    String jobId) {

    GetExportJobResponse getExportJobResult;
    String status;

    try {
        // Checks the job status until the job completes or fails.
```

```
GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
    .jobId(jobId)
    .applicationId(applicationId)
    .build();

do {
    getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
    status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
    System.out.format("Export job %s . . .\n", status);
    TimeUnit.SECONDS.sleep(3);

} while (!status.equals("COMPLETED") && !status.equals("FAILED"));

if (status.equals("COMPLETED")) {
    System.out.println("Finished exporting endpoints.");
} else {
    System.err.println("Failed to export endpoints.");
    System.exit(1);
}

} catch (PinpointException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Download files from an Amazon S3 bucket and write them to the path location.
public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

    String newPath;
    try {
        for (String key : objectKeys) {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(s3BucketName)
                .key(key)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
```

```
        String fileSuffix = new
SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
        newPath = path + fileSuffix + ".gz";
        File myFile = new File(newPath);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
    }
    System.out.println("Download finished.");

} catch (S3Exception | NullPointerException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateExportJob](#) à la section Référence des AWS SDK for Java 2.x API.

CreateImportJob

L'exemple de code suivant montre comment utiliser `CreateImportJob`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Importer un segment.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <bucket> <key> <roleArn>\s

            Where:
                appId - The application ID to create a segment for.
                bucket - The name of the Amazon S3 bucket that contains the
segment definitons.
                key - The key of the S3 object.
                roleArn - ARN of the role that allows Amazon Pinpoint to
access S3. You need to set trust management for this to work. See https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_policies\_elements\_principal.html
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
import job status.");
    }
}
```

```
        System.out.println("See application " + response.jobStatus() + " for import
job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
        String appId,
        String bucket,
        String key,
        String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                .defineSegment(true)
                .registerEndpoints(true)
                .roleArn(roleArn)
                .format(Format.JSON)
                .s3Url("s3://" + bucket + "/" + key)
                .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                .importJobRequest(importRequest)
                .applicationId(appId)
                .build();

            CreateImportJobResponse jobResponse =
client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateImportJob](#) à la section Référence des AWS SDK for Java 2.x API.

CreateSegment

L'exemple de code suivant montre comment utiliser `CreateSegment`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSegment {
    public static void main(String[] args) {
        final String usage = ""

                                Usage:  <appId>
```



```

        Where:
            appId - The application ID to create a segment
for.

        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        SegmentResponse result = createSegment(pinpoint, appId);
        System.out.println("Segment " + result.name() + " created.");
        System.out.println(result.segmentType());
        pinpoint.close();
    }

    public static SegmentResponse createSegment(PinpointClient client, String
appId) {
        try {
            Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();

            segmentAttributes.put("Team", AttributeDimension.builder()
                .attributeType(AttributeType.INCLUSIVE)
                .values("Lakers")
                .build());

            RecencyDimension recencyDimension =
RecencyDimension.builder()
                .duration("DAY_30")
                .recencyType("ACTIVE")
                .build();

            SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
                .recency(recencyDimension)
                .build();

```

```
SegmentDemographics segmentDemographics =
SegmentDemographics
    .builder()
    .build();

SegmentLocation segmentLocation = SegmentLocation
    .builder()
    .build();

SegmentDimensions dimensions = SegmentDimensions
    .builder()
    .attributes(segmentAttributes)
    .behavior(segmentBehaviors)
    .demographic(segmentDemographics)
    .location(segmentLocation)
    .build();

WriteSegmentRequest writeSegmentRequest =
WriteSegmentRequest.builder()
    .name("MySegment")
    .dimensions(dimensions)
    .build();

CreateSegmentRequest createSegmentRequest =
CreateSegmentRequest.builder()
    .applicationId(appId)
    .writeSegmentRequest(writeSegmentRequest)
    .build();

CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
    System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
    System.out.println("Done");
    return createSegmentResult.segmentResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateSegment](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteApp

L'exemple de code suivant montre comment utiliser `DeleteApp`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimer une application.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = ""

                Usage: <appId>

                Where:
                appId - The ID of the application to delete.
    }
}
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String appId = args[0];
    System.out.println("Deleting an application with ID: " + appId);
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deletePinApp(pinpoint, appId);
    System.out.println("Done");
    pinpoint.close();
}

public static void deletePinApp(PinpointClient pinpoint, String appId) {
    try {
        DeleteAppRequest appRequest = DeleteAppRequest.builder()
            .applicationId(appId)
            .build();

        DeleteAppResponse result = pinpoint.deleteApp(appRequest);
        String appName = result.applicationResponse().name();
        System.out.println("Application " + appName + " has been deleted.");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteApp](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteEndpoint

L'exemple de code suivant montre comment utiliser `DeleteEndpoint`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprime un point de terminaison.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = ""

                Usage:  <appName> <endpointId >

                Where:
                    appId - The id of the application to delete.
                    endpointId - The id of the endpoint to delete.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
    }
}
```

```
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deletePinEndpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    public static void deletePinEndpoint(PinpointClient pinpoint, String appId,
String endpointId) {
        try {
            DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpointId)
                .build();

            DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
            String id = result.endpointResponse().id();
            System.out.println("The deleted endpoint id " + id);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteEndpoint](#) à la section Référence des AWS SDK for Java 2.x API.

GetEndpoint

L'exemple de code suivant montre comment utiliser `GetEndpoint`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <endpoint>

            Where:
                appId - The ID of the application to delete.
                endpoint - The ID of the endpoint.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String appId = args[0];
String endpoint = args[1];
System.out.println("Looking up an endpoint point with ID: " + endpoint);
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

lookupPinpointEndpoint(pinpoint, appId, endpoint);
pinpoint.close();
}

public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
String endpoint) {
    try {
        GetEndpointRequest appRequest = GetEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpoint)
            .build();

        GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
        EndpointResponse endResponse = result.endpointResponse();

        // Uses the Google Gson library to pretty print the endpoint JSON.
        Gson gson = new GsonBuilder()
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
            .setPrettyPrinting()
            .create();

        String endpointJson = gson.toJson(endResponse);
        System.out.println(endpointJson);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetEndpoint](#) à la section Référence des AWS SDK for Java 2.x API.

GetSegments

L'exemple de code suivant montre comment utiliser `GetSegments`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertorier les segments.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = ""

                Usage:  <appId>

                Where:
                    appId - The ID of the application that contains a segment.

                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String appId = args[0];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    listSegs(pinpoint, appId);
    pinpoint.close();
}

public static void listSegs(PinpointClient pinpoint, String appId) {
    try {
        GetSegmentsRequest request = GetSegmentsRequest.builder()
            .applicationId(appId)
            .build();

        GetSegmentsResponse response = pinpoint.getSegments(request);
        List<SegmentResponse> segments = response.segmentsResponse().item();
        for (SegmentResponse segment : segments) {
            System.out
                .println("Segment " + segment.id() + " " + segment.name() +
                    " " + segment.lastModifiedDate());
        }


    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetSegments](#) à la section Référence des AWS SDK for Java 2.x API.

GetSmsChannel

L'exemple de code suivant montre comment utiliser `GetSmsChannel`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = ""

                Usage: CreateChannel <appId>

                Where:
                appId - The name of the application whose channel is updated.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
```

```
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

SMSChannelResponse getResponse = getSmsChannel(pinpoint, appId);
toggleSmsChannel(pinpoint, appId, getResponse);
pinpoint.close();
}

private static SMSChannelResponse getSmsChannel(PinpointClient client, String
appId) {
    try {
        GetSmsChannelRequest request = GetSmsChannelRequest.builder()
            .applicationId(appId)
            .build();

        SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
        System.out.println("Channel state is " + response.enabled());
        return response;

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
    boolean enabled = !getResponse.enabled();
    try {
        SMSChannelRequest request = SMSChannelRequest.builder()
            .enabled(enabled)
            .build();

        UpdateSmsChannelRequest updateRequest =
UpdateSmsChannelRequest.builder()
            .smsChannelRequest(request)
            .applicationId(appId)
            .build();

        UpdateSmsChannelResponse result =
client.updateSmsChannel(updateRequest);
    }
}
```

```
        System.out.println("Channel state: " +
result.smsChannelResponse().enabled());

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetSmsChannel](#) à la section Référence des AWS SDK for Java 2.x API.

GetUserEndpoints

L'exemple de code suivant montre comment utiliser `GetUserEndpoints`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
*/
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <applicationId> <userId>

            Where:
                applicationId - The ID of the Amazon Pinpoint application that
has the endpoint.
                userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String applicationId = args[0];
        String userId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllEndpoints(pinpoint, applicationId, userId);
        pinpoint.close();
    }

    public static void listAllEndpoints(PinpointClient pinpoint,
        String applicationId,
        String userId) {

        try {
            GetUserEndpointsRequest endpointsRequest =
GetUserEndpointsRequest.builder()
                .userId(userId)
                .applicationId(applicationId)
                .build();

            GetUserEndpointsResponse response =
pinpoint.getUserEndpoints(endpointsRequest);
            List<EndpointResponse> endpoints = response.endpointsResponse().item();

            // Display the results.
            for (EndpointResponse endpoint : endpoints) {
```

```
        System.out.println("The channel type is: " +
            endpoint.channelType());
        System.out.println("The address is " + endpoint.address());
    }

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetUserEndpoints](#) à la section Référence des AWS SDK for Java 2.x API.

SendMessage

L'exemple de code suivant montre comment utiliser `SendMessage`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoyer un e-mail.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessageRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
```

```
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessage {

    // The character encoding the you want to use for the subject line and
    // message body of the email.
    public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
    content.
    static final String body = """"
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
        for Java 2.x

        """;

    public static void main(String[] args) {
        final String usage = """"

            Usage:    <subject> <appId> <senderAddress>
<toAddress>

            Where:
                subject - The email subject to use.
                senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
```



```
        toAddress - The to address. This address has to be verified in Amazon
        Pinpoint in the region you're using to send email\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendEmail(pinpoint, subject, senderAddress, toAddress);
    System.out.println("Email was sent");
    pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
```

```

        .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();

        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

Envoyez un e-mail avec les valeurs CC.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessageCC {

    // The body of the email.

```

```
static final String body = """"
    Amazon Pinpoint test (AWS SDK for Java 2.x)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK
for Java 2.x

    """";
public static void main(String[] args) {
    final String usage = """"

        Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

        Where:
            subject - The email subject to use.
            senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
            toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
            ccAddress - The CC address.
        """";

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    String ccAddress = args[3];

    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    ArrayList<String> ccList = new ArrayList<>();
    ccList.add(ccAddress);
    sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
    pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
```

```
try {
    Content content = Content.builder()
        .data(body)
        .build();

    Body messageBody = Body.builder()
        .text(content)
        .build();

    Message message = Message.builder()
        .body(messageBody)
        .subject(Content.builder().data(subject).build())
        .build();

    Destination destination = Destination.builder()
        .toAddresses(toAddress)
        .ccAddresses(ccAddresses)
        .build();

    EmailContent emailContent = EmailContent.builder()
        .simple(message)
        .build();

    SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
        .fromEmailAddress(senderAddress)
        .destination(destination)
        .content(emailContent)
        .build();

    pinpointEmailClient.sendEmail(sendEmailRequest);
    System.out.println("Message Sent");

} catch (PinpointException e) {
    // Handle exception
    e.printStackTrace();
}
}
```

Envoyer un SMS.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessage {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber>\s

            Where:
```

```

        message - The body of the message to send.
        appId - The Amazon Pinpoint project/application ID
to use when you send this message.
        originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
        destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String appId = args[1];
    String originationNumber = args[2];
    String destinationNumber = args[3];
    System.out.println("Sending a message");
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
    pinpoint.close();
}

public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
    String originationNumber,
    String destinationNumber) {
    try {
        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
        AddressConfiguration addConfig =
AddressConfiguration.builder()
            .channelType(ChannelType.SMS)
            .build();

        addressMap.put(destinationNumber, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()

```

```

        .body(message)
        .messageType(messageType)
        .originationNumber(originationNumber)
        .senderId(senderId)
        .keyword(registeredKeyword)
        .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
        .smsMessage(smsMessage)
        .build();

        MessageRequest msgReq = MessageRequest.builder()
        .addresses(addressMap)
        .messageConfiguration(direct)
        .build();

        // create a SendMessagesRequest object
        SendMessagesRequest request = SendMessagesRequest.builder()
        .applicationId(appId)
        .messageRequest(msgReq)
        .build();

        SendMessagesResponse response =
pinpoint.sendMessage(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

Envoyer des SMS par lots.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber> <destinationNumber1>\s

            Where:
```



```

        message - The body of the message to send.
        appId - The Amazon Pinpoint project/application ID
to use when you send this message.
        originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
        destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
        destinationNumber1 - The second recipient's phone
number. For best results, you should specify the phone number in E.164 format (for
example, +1-555-555-5654).\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String appId = args[1];
    String originationNumber = args[2];
    String destinationNumber = args[3];
    String destinationNumber1 = args[4];
    System.out.println("Sending a message");
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
    pinpoint.close();
}

public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
        String originationNumber,
        String destinationNumber, String destinationNumber1) {
    try {
        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
        AddressConfiguration addConfig =
AddressConfiguration.builder()
            .channelType(ChannelType.SMS)

```

```
        .build());

        // Add an entry to the Map object for each number to whom
you want to send a
        // message.
        addressMap.put(destinationNumber, addConfig);
        addressMap.put(destinationNumber1, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();

        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // Create a SendMessagesRequest object.
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)
            .messageRequest(msgReq)
            .build();

        SendMessagesResponse response =
pinpoint.sendMessage(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [SendMessages](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateEndpoint

L'exemple de code suivant montre comment utiliser `UpdateEndpoint`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.pinpoint.PinpointClient;  
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;  
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;  
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;  
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;  
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;  
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;  
import software.amazon.awssdk.services.pinpoint.model.PinpointException;  
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;  
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;  
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.List;  
import java.util.UUID;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Date;  
  
/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage: <appId>

            Where:
                appId - The ID of the application to create an endpoint for.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        EndpointResponse response = createEndpoint(pinpoint, appId);
        System.out.println("Got Endpoint: " + response.id());
        pinpoint.close();
    }

    public static EndpointResponse createEndpoint(PinpointClient client, String
appId) {
        String endpointId = UUID.randomUUID().toString();
        System.out.println("Endpoint ID: " + endpointId);

        try {
            EndpointRequest endpointRequest = createEndpointRequestData();
            UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpointId)
```

```
        .endpointRequest(endpointRequest)
        .build();

        UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
        System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

        GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
        .applicationId(appId)
        .endpointId(endpointId)
        .build();

        GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
        System.out.println(getEndpointResponse.endpointResponse().address());

System.out.println(getEndpointResponse.endpointResponse().channelType());

System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
        System.out.println(getEndpointResponse.endpointResponse().requestId());
        System.out.println(getEndpointResponse.endpointResponse().user());

        return getEndpointResponse.endpointResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static EndpointRequest createEndpointRequestData() {
    try {
        List<String> favoriteTeams = new ArrayList<>();
        favoriteTeams.add("Lakers");
        favoriteTeams.add("Warriors");
        HashMap<String, List<String>> customAttributes = new HashMap<>();
        customAttributes.put("team", favoriteTeams);

        EndpointDemographic demographic = EndpointDemographic.builder()
        .appVersion("1.0")
```

```
        .make("apple")
        .model("iPhone")
        .modelVersion("7")
        .platform("ios")
        .platformVersion("10.1.1")
        .timezone("America/Los_Angeles")
        .build();

    EndpointLocation location = EndpointLocation.builder()
        .city("Los Angeles")
        .country("US")
        .latitude(34.0)
        .longitude(-118.2)
        .postalCode("90068")
        .region("CA")
        .build();

    Map<String, Double> metrics = new HashMap<>();
    metrics.put("health", 100.00);
    metrics.put("luck", 75.00);

    EndpointUser user = EndpointUser.builder()
        .userId(UUID.randomUUID().toString())
        .build();

    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());

    return EndpointRequest.builder()
        .address(UUID.randomUUID().toString())
        .attributes(customAttributes)
        .channelType("APNS")
        .demographic(demographic)
        .effectiveDate(nowAsISO)
        .location(location)
        .metrics(metrics)
        .optOut("NONE")
        .requestId(UUID.randomUUID().toString())
        .user(user)
        .build();

    } catch (PinpointException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateEndpoint](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'API SMS et vocales Amazon Pinpoint à l'aide du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide de l'API SMS et voix Amazon Pinpoint.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

SendVoiceMessage

L'exemple de code suivant montre comment utiliser `SendVoiceMessage`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
    list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";
```



```

// The content of the message. This example uses SSML to customize and
control
// certain aspects of the message, such as by adding pauses and changing
// phonation. The message can't contain any line breaks.
static final String ssmlMessage = "<speak>This is a test message sent from "
    + "<emphasis>Amazon Pinpoint</emphasis> "
    + "using the <break strength='weak'>AWS "
    + "SDK for Java. "
    + "<amazon:effect phonation='soft'>Thank "
    + "you for listening.</amazon:effect></speak>";

public static void main(String[] args) {

    final String usage = ""

        Usage:  <originationNumber> <destinationNumber>\s

        Where:
            originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
            destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s

        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String originationNumber = args[0];
    String destinationNumber = args[1];
    System.out.println("Sending a voice message");

    // Set the content type to application/json.
    List<String> listVal = new ArrayList<>();
    listVal.add("application/json");
    Map<String, List<String>> values = new HashMap<>();
    values.put("Content-Type", listVal);

    ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
        .headers(values)

```

```
        .build();

        PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
            .overrideConfiguration(config2)
            .region(Region.US_EAST_1)
            .build();

        sendVoiceMsg(client, originationNumber, destinationNumber);
        client.close();
    }

    public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
        String destinationNumber) {
        try {
            SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                .languageCode(languageCode)
                .text(ssmlMessage)
                .voiceId(voiceName)
                .build();

            VoiceMessageContent content = VoiceMessageContent.builder()
                .ssmlMessage(ssmlMessageType)
                .build();

            SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
                .destinationPhoneNumber(destinationNumber)
                .originationPhoneNumber(originationNumber)
                .content(content)
                .build();

            client.sendVoiceMessage(voiceMessageRequest);
            System.out.println("The message was sent successfully.");

        } catch (PinpointSmsVoiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SendVoiceMessage](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'Amazon Polly utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Polly.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

DescribeVoices

L'exemple de code suivant montre comment utiliser `DescribeVoices`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
```

```
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                .languageCode("en-US")
                .build();

            DescribeVoicesResponse enUsVoicesResult =
polly.describeVoices(voicesRequest);
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeVoices](#) à la section Référence des AWS SDK for Java 2.x API.

ListLexicons

L'exemple de code suivant montre comment utiliser `ListLexicons`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listLexicons(polly);
        polly.close();
    }
}
```

```
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest = ListLexiconsRequest.builder()
                .build();

            ListLexiconsResponse listLexiconsResult =
client.listLexicons(listLexiconsRequest);
            List<LexiconDescription> lexiconDescription =
listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
lexDescription.name());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListLexicons](#) à la section Référence des AWS SDK for Java 2.x API.

SynthesizeSpeech

L'exemple de code suivant montre comment utiliser SynthesizeSpeech.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
        built this working demo " +
        " of Amazon Polly in Java Version 2. Have fun building voice enabled
        apps with Amazon Polly (that's me!), and always "
        +
        " look at the AWS website for tips and tricks on using Amazon Polly and
        other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
```

```

        DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
            .engine("standard")
            .build();

        DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
        Voice voice = describeVoicesResult.voices().stream()
            .filter(v -> v.name().equals("Joanna"))
            .findFirst()
            .orElseThrow(() -> new RuntimeException("Voice not found"));
        InputStream stream = synthesize(polly, SAMPLE, voice, OutputFormat.MP3);
        AdvancedPlayer player = new AdvancedPlayer(stream,

javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
        player.setPlayBackListener(new PlaybackListener() {
            public void playbackStarted(PlaybackEvent evt) {
                System.out.println("Playback started");
                System.out.println(SAMPLE);
            }

            public void playbackFinished(PlaybackEvent evt) {
                System.out.println("Playback finished");
            }
        });

        // play it!
        player.play();

    } catch (PollyException | JavaLayerException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static InputStream synthesize(PollyClient polly, String text, Voice
voice, OutputFormat format)
    throws IOException {
    SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
        .text(text)
        .voiceId(voice.id())
        .outputFormat(format)
        .build();

```



```
        ResponseInputStream<SynthesizeSpeechResponse> synthRes =
polly.synthesizeSpeech(synthReq);
        return synthRes;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SynthesizeSpeech](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples Amazon RDS utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon RDS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Hello Amazon RDS

Les exemples de code suivants montrent comment bien démarrer avec Amazon RDS.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
instance.endpoint().address());
            }

            } catch (RdsException e) {
                System.out.println(e.getLocalizedMessage());
                System.exit(1);
            }
        }
    }
}
```

- Pour plus d'informations sur l'API, consultez [DescribeDBInstances](#) dans la Référence d'API AWS SDK for Java 2.x .

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateDBInstance

L'exemple de code suivant montre comment utiliser CreateDBInstance.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-services-use-secrets\_RS.html
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <dbName> <secretName>

            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                dbName - The database name.\s
                secretName - The name of the AWS Secrets Manager secret that
contains the database credentials."
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
    }
}
```

```
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
        String dbInstanceIdentifier,
        String dbName,
        String userName,
        String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
```

```
        .dbName(dbName)
        .engine("mysql")
        .dbInstanceClass("db.m4.large")
        .engineVersion("8.0")
        .storageType("standard")
        .masterUsername(userName)
        .masterUserPassword(userPassword)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        // Loop until the cluster is ready.
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available"))
                    instanceReady = true;
                else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
    }
}
```

```
        }
    }
    System.out.println("Database instance is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus d'informations sur l'API, consultez [CreateDBInstance](#) dans AWS SDK for Java 2.x API Reference.

CreateDBParameterGroup

L'exemple de code suivant montre comment utiliser `CreateDBParameterGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());
    }
}
```

```
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, voir [CreateDB ParameterGroup](#) dans la référence des AWS SDK for Java 2.x API.

CreateDBSnapshot

L'exemple de code suivant montre comment utiliser CreateDBSnapshot.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Create an Amazon RDS snapshot.  
public static void createSnapshot(RdsClient rdsClient, String  
dbInstanceIdentifier, String dbSnapshotIdentifier) {  
    try {  
        CreateDbSnapshotRequest snapshotRequest =  
CreateDbSnapshotRequest.builder()  
            .dbInstanceIdentifier(dbInstanceIdentifier)  
            .dbSnapshotIdentifier(dbSnapshotIdentifier)  
            .build();  
  
        CreateDbSnapshotResponse response =  
rdsClient.createDBSnapshot(snapshotRequest);  
        System.out.println("The Snapshot id is " +  
response.dbSnapshot().dbiResourceId());  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```



```
    }  
}
```

- Pour plus d'informations sur l'API, consultez [CreateDBSnapshot](#) dans la Référence d'API AWS SDK for Java 2.x .

DeleteDBInstance

L'exemple de code suivant montre comment utiliser `DeleteDBInstance`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rds.RdsClient;  
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;  
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;  
import software.amazon.awssdk.services.rds.model.RdsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DeleteDBInstance {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
                <dbInstanceIdentifier>\s  
  
            Where:
```

```
        dbInstanceIdentifier - The database instance identifier\s
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbInstanceIdentifier = args[0];
    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();

    deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
    rdsClient.close();
}

public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez [DeleteDBInstance](#) dans la Référence d'API AWS SDK for Java 2.x .

DeleteDBParameterGroup

L'exemple de code suivant montre comment utiliser `DeleteDBParameterGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
        }
    }
}
```

```
        index++;
    }
}

// Delete the para group.
DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .build();

rdsClient.deleteDBParameterGroup(parameterGroupRequest);
System.out.println(dbGroupName + " was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, voir [DeleteDB ParameterGroup dans la référence](#) des AWS SDK for Java 2.x API.

DescribeAccountAttributes

L'exemple de code suivant montre comment utiliser `DescribeAccountAttributes`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAccountAttributes](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeDBEngineVersions

L'exemple de code suivant montre comment utiliser `DescribeDBEngineVersions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, voir [DescribeDB EngineVersions dans le Guide](#) de référence des AWS SDK for Java 2.x API.

DescribeDBInstances

L'exemple de code suivant montre comment utiliser `DescribeDBInstances`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }
}
```

```
public static void describeInstances(RdsClient rdsClient) {
    try {
        DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
        List<DBInstance> instanceList = response.dbInstances();
        for (DBInstance instance : instanceList) {
            System.out.println("Instance ARN is: " + instance.dbInstanceArn());
            System.out.println("The Engine is " + instance.engine());
            System.out.println("Connection endpoint is" +
instance.endpoint().address());
        }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus d'informations sur l'API, consultez [DescribeDBInstances](#) dans la Référence d'API AWS SDK for Java 2.x .

DescribeDBParameterGroups

L'exemple de code suivant montre comment utiliser `DescribeDBParameterGroups`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
```



```
        .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
            System.out.println("The group description is " +
group.description());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, voir [DescribeDB ParameterGroups dans le Guide](#) de référence des AWS SDK for Java 2.x API.

DescribeDBParameters

L'exemple de code suivant montre comment utiliser `DescribeDBParameters`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
```

```
        .dbParameterGroupName(dbGroupName)
        .build();
    } else {
        dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .source("user")
            .build();
    }

    DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
    List<Parameter> dbParameters = response.parameters();
    String paraName;
    for (Parameter para : dbParameters) {
        // Only print out information about either auto_increment_offset or
        // auto_increment_increment.
        paraName = para.parameterName();
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
para.parameterValue());
            System.out.println("*** The parameter data type is " +
para.dataType());
            System.out.println("*** The parameter description is " +
para.description());
            System.out.println("*** The parameter allowed values is " +
para.allowedValues());
        }
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus d'informations sur l'API, consultez [DescribeDBParameters](#) dans la Référence d'API AWS SDK for Java 2.x .

DescribeOrderableDBInstanceOptions

L'exemple de code suivant montre comment utiliser `DescribeOrderableDBInstanceOptions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, voir [DescribeOrderableDB InstanceOptions](#) dans le Guide de référence des AWS SDK for Java 2.x API.

GenerateRDSAuthToken

L'exemple de code suivant montre comment utiliser `GenerateRDSAuthToken`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez la [RdsUtilities](#) classe pour générer un jeton d'authentification.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <masterUsername>

            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUsername - The master user name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
            masterUsername);
        System.out.println("The token response is " + token);
    }
}
```

```
public static String getAuthToken(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUsername) {

    RdsUtilities utilities = rdsClient.utilities();
    try {
        GenerateAuthenticationTokenRequest tokenRequest =
GenerateAuthenticationTokenRequest.builder()
            .credentialsProvider(ProfileCredentialsProvider.create())
            .username(masterUsername)
            .port(3306)
            .hostname(dbInstanceIdentifier)
            .build();

        return utilities.generateAuthenticationToken(tokenRequest);

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, voir [GenerateRds AuthToken](#) dans la référence des AWS SDK for Java 2.x API.

ModifyDBInstance

L'exemple de code suivant montre comment utiliser `ModifyDBInstance`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
```

```
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUserPassword - The updated password that corresponds to
the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master password.

```

```

        ModifyDbInstanceRequest modifyDbInstanceRequest =
ModifyDbInstanceRequest.builder()
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .publiclyAccessible(true)
    .masterUserPassword(masterUserPassword)
    .build();

        ModifyDbInstanceResponse instanceResponse =
rdsClient.modifyDBInstance(modifyDbInstanceRequest);
        System.out.println("The ARN of the modified database is: " +
instanceResponse.dbInstance().dbInstanceArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
}

```

- Pour plus d'informations sur l'API, consultez [ModifyDBInstance](#) dans la Référence d'API AWS SDK for Java 2.x .

ModifyDBParameterGroup

L'exemple de code suivant montre comment utiliser `ModifyDBParameterGroup`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")

```

```
        .parameterValue("5")
        .build();

    List<Parameter> paraList = new ArrayList<>();
    paraList.add(parameter1);
    ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .parameters(paraList)
        .build();

    ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
    System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, voir [ModifyDB ParameterGroup dans la référence](#) des AWS SDK for Java 2.x API.

RebootDBInstance

L'exemple de code suivant montre comment utiliser `RebootDBInstance`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
```



```
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier>\s

                Where:
                dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
RebootDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .build();
```

```
        RebootDbInstanceResponse instanceResponse =
rdsClient.rebootDBInstance(rebootDbInstanceRequest);
        System.out.print("The database " +
instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez [RebootDBInstance](#) dans la Référence d'API AWS SDK for Java 2.x .

Scénarios

Démarrage avec les instances de base de données

L'exemple de code suivant illustre comment :

- Créez un groupe de paramètres de bases de données personnalisé et définissez des valeurs pour les paramètres.
- Créez une instance de base de données configurée pour utiliser le groupe de paramètres. L'instance de base de données contient également une base de données.
- Prenez un instantané de l'instance.
- Supprimez l'instance et le groupe de paramètres.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez plusieurs opérations.

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-services-use-secrets\_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySQL database and uses
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
                <dbName> <dbSnapshotIdentifier> <secretName>
```

```
        Where:
            dbGroupName - The database group name.\s
            dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
            dbInstanceIdentifier - The database instance identifier\s
            dbName - The database name.\s
            dbSnapshotIdentifier - The snapshot identifier.\s
            secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"
        """;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbGroupName = args[0];
    String dbParameterGroupFamily = args[1];
    String dbInstanceIdentifier = args[2];
    String dbName = args[3];
    String dbSnapshotIdentifier = args[4];
    String secretName = args[5];

    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    String masterUsername = user.getUsername();
    String masterUserPassword = user.getPassword();

    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();
    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon RDS example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Return a list of the available DB engines");
    describeDBEngines(rdsClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```
System.out.println("2. Create a custom parameter group");
createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbParameterGroups(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbParameters(rdsClient, dbGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBParas(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated value");
describeDbParameters(rdsClient, dbGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get a list of micro instance classes available for
the selected engine");
getMicroInstances(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "9. Create an RDS database instance that contains a MySql database
and uses the parameter group");
String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
    masterUserPassword);
System.out.println("The ARN of the new database is " + dbARN);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("10. Wait for DB instance to be ready");
waitForInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a snapshot of the DB instance");
createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB snapshot to be ready");
waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Delete the DB instance");
deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Delete the parameter group");
deleteParaGroup(rdsClient, dbGroupName, dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed.");
System.out.println(DASHES);

rdsClient.close();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

public static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
```

```
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
        throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }
}
```



```
        // Delete the para group.
        DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .build();

        rdsClient.deleteDBParameterGroup(parameterGroupRequest);
        System.out.println(dbGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Delete the DB instance.
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the snapshot instance is available.
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
    String dbSnapshotIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
```

```
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
        .dbSnapshotIdentifier(dbSnapshotIdentifier)
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .build();

        while (!snapshotReady) {
            DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
            List<DBSnapshot> snapshotList = response.dbSnapshots();
            for (DBSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }

        System.out.println("The Snapshot is available!");
    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());
    }
}
```

```
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
    String dbGroupName,
    String dbInstanceIdentifier,
    String dbName,
    String masterUsername,
    String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .dbParameterGroupName(dbGroupName)
            .engine("mysql")
            .dbInstanceClass("db.m4.large")
            .engineVersion("8.0")
            .storageType("standard")
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.println("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }

    return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()
            .engine("mysql")
```

```
        .build();

        DescribeOrderableDbInstanceOptionsResponse response = rdsClient
            .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
        List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
        for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
    {
            System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
            System.out.println("The engine description is " +
dbInstanceOption.engine());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
}

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }
    }
}
```

```
    }

    DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
    List<Parameter> dbParameters = response.parameters();
    String paraName;
    for (Parameter para : dbParameters) {
        // Only print out information about either auto_increment_offset or
        // auto_increment_increment.
        paraName = para.parameterName();
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
para.parameterValue());
            System.out.println("*** The parameter data type is " +
para.dataType());
            System.out.println("*** The parameter description is " +
para.description());
            System.out.println("*** The parameter allowed values is " +
para.allowedValues());
        }
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group : groups) {
```

```
        System.out.println("The group name is " +
group.dbParameterGroupName());
        System.out.println("The group description is " +
group.description());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
```



```
List<DBEngineVersion> engines = response.dbEngineVersions();

// Get all DBEngineVersion objects.
for (DBEngineVersion engineOb : engines) {
    System.out.println("The name of the DB parameter group family for
the database engine is "
        + engineOb.dbParameterGroupFamily());
    System.out.println("The name of the database engine " +
engineOb.engine());
    System.out.println("The version number of the database engine " +
engineOb.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateDBInstance](#)
 - [Créer une base de données ParameterGroup](#)
 - [CreateDBSnapshot](#)
 - [DeleteDBInstance](#)
 - [Supprimer B ParameterGroup](#)
 - [Décrit B EngineVersions](#)
 - [DescribeDBInstances](#)
 - [Décrit B ParameterGroups](#)
 - [DescribeDBParameters](#)
 - [DescribeDBSnapshots](#)
 - [DescribeOrderableDB InstanceOptions](#)
 - [Modifier la base de données ParameterGroup](#)

Exemples d'Amazon Redshift utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Redshift.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Amazon Redshift

Les exemples de code suivants montrent comment commencer à utiliser Amazon Redshift.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class HelloRedshift {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        listClustersPaginator(redshiftClient);
    }

    public static void listClustersPaginator(RedshiftClient redshiftClient) {
        DescribeClustersIterable clustersIterable =
redshiftClient.describeClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.clusters().stream())
            .forEach(cluster -> System.out
                .println(" Cluster identifier: " + cluster.clusterIdentifier() + "
status = " + cluster.clusterStatus()));
    }
}
```

- Pour plus de détails sur l'API, consultez [DescribeClusters](#) dans AWS SDK for Java 2.x la référence des API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateCluster

L'exemple de code suivant montre comment utiliser CreateCluster.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez le cluster .

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCluster](#) à la section Référence des AWS SDK for Java 2.x API.

CreateTable

L'exemple de code suivant montre comment utiliser CreateTable.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
    try {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .dbUser(userName)
        .database(databaseName)
        .sql("CREATE TABLE Movies ("
            + "id INT PRIMARY KEY, "
            + "title VARCHAR(100), "
            + "year INT)")
        .build();

        redshiftDataClient.executeStatement(createTableRequest);
        System.out.println("Table created: Movies");

    } catch (RedshiftDataException e) {
        System.err.println("Error creating table: " + e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTable](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteCluster

L'exemple de code suivant montre comment utiliser DeleteCluster.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez le cluster.

```
public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
        .clusterIdentifier(clusterId)
        .skipFinalClusterSnapshot(true)
        .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is " +
response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCluster](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeClusters

L'exemple de code suivant montre comment utiliser `DescribeClusters`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Décrivez le cluster.

```
public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
    String clusterReadyStr;
    System.out.println("Waiting for cluster to become available. This may take a
few mins.");
    try {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();
        long startTime = System.currentTimeMillis();

        // Loop until the cluster is ready.
        while (!clusterReady) {
            DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
            List<Cluster> clusterList = clusterResponse.clusters();
            for (Cluster cluster : clusterList) {
                clusterReadyStr = cluster.clusterStatus();
                if (clusterReadyStr.contains("available"))
                    clusterReady = true;
                else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;

                    System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
                    TimeUnit.SECONDS.sleep(5);
                }
            }
        }
    }
}
```

```
        }
    }

    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
    long elapsedSeconds = elapsedTimeMillis / 1000;
    long minutes = elapsedSeconds / 60;
    long seconds = elapsedSeconds % 60;

    System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

    } catch (RedshiftException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeClusters](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeStatement

L'exemple de code suivant montre comment utiliser `DescribeStatement`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void checkStatement(RedshiftDataClient redshiftDataClient, String
sqlId) {
    try {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();
```



```
String status;
while (true) {
    DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
    status = response.statusAsString();
    System.out.println("..." + status);

    if (status.compareTo("FAILED") == 0 ) {
        System.out.println("The Query Failed. Ending program");
        System.exit(1);

    } else if (status.compareTo("FINISHED") == 0) {
        break;
    }
    TimeUnit.SECONDS.sleep(1);
}

System.out.println("The statement is finished!");

} catch (RedshiftDataException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeStatement](#) à la section Référence des AWS SDK for Java 2.x API.

GetStatementResult

L'exemple de code suivant montre comment utiliser `GetStatementResult`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Vérifiez le résultat de la déclaration.

```
public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
        GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
        redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
            .map(Field::stringValue)
            .filter(value -> value != null)
            .forEach(value -> System.out.println("The Movie title field is " +
value));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetStatementResult](#) à la section Référence des AWS SDK for Java 2.x API.

Insert

L'exemple de code suivant montre comment utiliser `Insert`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
        String title = currentNode.get("title").asText();

        // Use SqlParameter to avoid SQL injection.
        List<SqlParameter> parameterList = new ArrayList<>();
        String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";

        // Create the parameters.
        SqlParameter idParam = SqlParameter.builder()
            .name("id")
            .value(String.valueOf(t))
            .build();

        SqlParameter titleParam= SqlParameter.builder()
            .name("title")
            .value(title)
            .build();

        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
        parameterList.add(idParam);
        parameterList.add(titleParam);
        parameterList.add(yearParam);

        try {
            ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
```

```
        .clusterIdentifier(clusterId)
        .sql(sqlStatement)
        .database(databaseName)
        .dbUser(userName)
        .parameters(parameterList)
        .build();

        redshiftDataClient.executeStatement(insertStatementRequest);
        System.out.println("Inserted: " + title + " (" + year + ")");
        t++;

    } catch (RedshiftDataException e) {
        System.err.println("Error inserting data: " + e.getMessage());
        System.exit(1);
    }
}
System.out.println(t + " records were added to the Movies table. ");
}
```

- Pour plus de détails sur l'API, voir [Insérer](#) dans la référence AWS SDK for Java 2.x d'API.

ModifyCluster

L'exemple de code suivant montre comment utiliser `ModifyCluster`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Modifiez un cluster.

```
public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        ModifyClusterRequest modifyClusterRequest =
        ModifyClusterRequest.builder()
```

```
        .clusterIdentifier(clusterId)
        .preferredMaintenanceWindow("wed:07:30-wed:08:00")
        .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
            + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ModifyCluster](#) à la section Référence des AWS SDK for Java 2.x API.

Query

L'exemple de code suivant montre comment utiliser `Query`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Interrogez une table.

```
public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                       String database,
                                       String dbUser,
                                       int year,
                                       String clusterId) {
```

```
try {
    String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
    SqlParameter yearParam= SqlParameter.builder()
        .name("year")
        .value(String.valueOf(year))
        .build();

    ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .database(database)
        .dbUser(dbUser)
        .parameters(yearParam)
        .sql(sqlStatement)
        .build();

    ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
    return response.id();

} catch (RedshiftDataException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```


- Pour plus d'informations sur l'API, consultez [Requête](#) dans la référence d'API AWS SDK for Java 2.x .

Scénarios

Commencez avec Amazon Redshift

L'exemple de code suivant montre comment utiliser les tables, les éléments et les requêtes Amazon Redshift.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.Cluster;
import software.amazon.awssdk.services.redshift.model.CreateClusterRequest;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.DescribeClustersRequest;
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterRequest;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.RedshiftDataClient;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.Field;
import software.amazon.awssdk.services.redshiftdata.model.GetStatementResultRequest;
import
    software.amazon.awssdk.services.redshiftdata.model.GetStatementResultResponse;
import software.amazon.awssdk.services.redshiftdata.model.ListDatabasesRequest;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import software.amazon.awssdk.services.redshiftdata.model.SqlParameter;
import
    software.amazon.awssdk.services.redshiftdata.paginators.ListDatabasesIterable;
import com.fasterxml.jackson.core.JsonParser;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
 * 6. Inserts a specified number of records into the "Movies" table by reading the
 * Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
 * 11. If confirmed, deletes the specified Redshift cluster.
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <jsonFilePath>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
file in ../../../../resources/sample_files/movies.json)
                """;

        if (args.length != 1) {
```



```
        System.out.println(usage);
        System.exit(1);
    }

    String jsonFilePath = args[0];
    String userName;
    String userPassword;
    String databaseName = "dev" ;
    Scanner scanner = new Scanner(System.in);

    Region region = Region.US_EAST_1;
    RedshiftClient redshiftClient = RedshiftClient.builder()
        .region(region)
        .build();

    RedshiftDataClient redshiftDataClient = RedshiftDataClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon Redshift SDK Getting Started
scenario.");
    System.out.println("""
    This Java program demonstrates how to interact with Amazon Redshift by using
the AWS SDK for Java (v2).\s
    Amazon Redshift is a fully managed, petabyte-scale data warehouse service
hosted in the cloud.

    The program's primary functionalities include cluster creation, verification
of cluster readiness,\s
    list databases, table creation, data population within the table, and
execution of SQL statements.
    Furthermore, it demonstrates the process of querying data from the Movie
table.\s

    Upon completion of the program, all AWS resources are cleaned up.
""");

    System.out.println("Lets get started...");
    System.out.println("Please enter your user name (default is awsuser)");
    String user = scanner.nextLine();
    userName = user.isEmpty() ? "awsuser" : user;
    System.out.println(DASHES);
```

```

        System.out.println("Please enter your user password (default is
        AwsUser1000)");
        String userpass = scanner.nextLine();
        userPassword = userpass.isEmpty() ? "AwsUser1000" : userpass;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A Redshift cluster refers to the collection of computing
        resources and storage that work together to process and analyze large volumes of
        data.");
        System.out.println("Enter a cluster id value (default is redshift-cluster-
        movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
        userClusterId;
        createCluster(redshiftClient, clusterId, userName, userPassword);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Wait until "+clusterId+" is available.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        waitForClusterReady(redshiftClient, clusterId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        String databaseInfo = ""
            When you created $clusteridD, the dev database is created by default and
            used in this scenario.\s

            To create a custom database, you need to have a CREATEDB privilege.\s
            For more information, see the documentation here: https://
            docs.aws.amazon.com/redshift/latest/dg/r\_CREATE\_DATABASE.html.
            """.replace("$clusteridD", clusterId);

        System.out.println(databaseInfo);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("List databases in "+clusterId);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();

```

```
listAllDatabases(redshiftDataClient, clusterId, userName, databaseName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now you will create a table named Movies.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
createTable(redshiftDataClient, clusterId, databaseName, userName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Populate the Movies table using the Movies.json file.");
System.out.println("Specify the number of records you would like to add to
the Movies Table.");
System.out.println("Please enter a value between 50 and 200.");
int numRecords;
do {
    System.out.print("Enter a value: ");
    while (!scanner.hasNextInt()) {
        System.out.println("Invalid input. Please enter a value between 50
and 200.");
        System.out.print("Enter a year: ");
        scanner.next();
    }
    numRecords = scanner.nextInt();
} while (numRecords < 50 || numRecords > 200);
popTable(redshiftDataClient, clusterId, databaseName, userName,
jsonFilePath, numRecords);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Query the Movies table by year. Enter a value between
2012-2014.");
int movieYear;
do {
    System.out.print("Enter a year: ");
    while (!scanner.hasNextInt()) {
        System.out.println("Invalid input. Please enter a valid year between
2012 and 2014.");
        System.out.print("Enter a year: ");
        scanner.next();
    }
    movieYear = scanner.nextInt();
    scanner.nextLine();
}
```

```
    } while (movieYear < 2012 || movieYear > 2014);

    String id = queryMoviesByYear(redshiftDataClient, databaseName, userName,
movieYear, clusterId);
    System.out.println("The identifier of the statement is " + id);
    checkStatement(redshiftDataClient, id);
    getResults(redshiftDataClient, id);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Now you will modify the Redshift cluster.");
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    modifyCluster(redshiftClient, clusterId);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Would you like to delete the Amazon Redshift cluster?
(y/n)");
    String delAns = scanner.nextLine().trim();
    if (delAns.equalsIgnoreCase("y")) {
        System.out.println("You selected to delete " +clusterId);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        deleteRedshiftCluster(redshiftClient, clusterId);
    } else {
        System.out.println("The "+clusterId +" was not deleted");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("This concludes the Amazon Redshift SDK Getting Started
scenario.");
    System.out.println(DASHES);
}

public static void listAllDatabases(RedshiftDataClient redshiftDataClient,
String clusterId, String dbUser, String database) {
    try {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();
```

```
        ListDatabasesIterable listDatabasesIterable =
redshiftDataClient.listDatabasesPaginator(databasesRequest);
        listDatabasesIterable.stream()
            .flatMap(r -> r.databases().stream())
            .forEach(db -> System.out
                .println("The database name is : " + db));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is " +
response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
```

```
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
        String title = currentNode.get("title").asText();

        // Use SqlParameter to avoid SQL injection.
        List<SqlParameter> parameterList = new ArrayList<>();
        String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";

        // Create the parameters.
        SqlParameter idParam = SqlParameter.builder()
            .name("id")
            .value(String.valueOf(t))
            .build();

        SqlParameter titleParam= SqlParameter.builder()
            .name("title")
            .value(title)
            .build();

        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
        parameterList.add(idParam);
        parameterList.add(titleParam);
        parameterList.add(yearParam);

        try {
            ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                .clusterIdentifier(clusterId)
                .sql(sqlStatement)
                .database(databaseName)
                .dbUser(userName)
                .parameters(parameterList)
                .build();

            redshiftDataClient.executeStatement(insertStatementRequest);
            System.out.println("Inserted: " + title + " (" + year + ")");
            t++;
        }
```

```
        } catch (RedshiftDataException e) {
            System.err.println("Error inserting data: " + e.getMessage());
            System.exit(1);
        }
    }
    System.out.println(t + " records were added to the Movies table. ");
}

public static void checkStatement(RedshiftDataClient redshiftDataClient, String
sqlId) {
    try {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        String status;
        while (true) {
            DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
            status = response.statusAsString();
            System.out.println("..." + status);

            if (status.compareTo("FAILED") == 0 ) {
                System.out.println("The Query Failed. Ending program");
                System.exit(1);
            }

            else if (status.compareTo("FINISHED") == 0) {
                break;
            }
            TimeUnit.SECONDS.sleep(1);
        }

        System.out.println("The statement is finished!");
    } catch (RedshiftDataException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
```

```
        ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
    .clusterIdentifier(clusterId)
    .preferredMaintenanceWindow("wed:07:30-wed:08:00")
    .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
            + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                        String database,
                                        String dbUser,
                                        int year,
                                        String clusterId) {

    try {
        String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam= SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
        return response.id();
    }
}
```



```
    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
            .map(Field::stringValue)
            .filter(value -> value != null)
            .forEach(value -> System.out.println("The Movie title field is " +
value));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
    String clusterReadyStr;
    System.out.println("Waiting for cluster to become available. This may take a
few mins.");
    try {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();
        long startTime = System.currentTimeMillis();
```

```
// Loop until the cluster is ready.
while (!clusterReady) {
    DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
    List<Cluster> clusterList = clusterResponse.clusters();
    for (Cluster cluster : clusterList) {
        clusterReadyStr = cluster.clusterStatus();
        if (clusterReadyStr.contains("available"))
            clusterReady = true;
        else {
            long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

            long elapsedSeconds = elapsedTimeMillis / 1000;
            long minutes = elapsedSeconds / 60;
            long seconds = elapsedSeconds % 60;

            System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
            TimeUnit.SECONDS.sleep(5);
        }
    }
}

long elapsedTimeMillis = System.currentTimeMillis() - startTime;
long elapsedSeconds = elapsedTimeMillis / 1000;
long minutes = elapsedSeconds / 60;
long seconds = elapsedSeconds % 60;

System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

} catch (RedshiftException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
    try {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
```

```
        .dbUser(userName)
        .database(databaseName)
        .sql("CREATE TABLE Movies ("
            + "id INT PRIMARY KEY, "
            + "title VARCHAR(100), "
            + "year INT)")
        .build();

        redshiftDataClient.executeStatement(createTableRequest);
        System.out.println("Table created: Movies");

    } catch (RedshiftDataException e) {
        System.err.println("Error creating table: " + e.getMessage());
        System.exit(1);
    }
}

public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [createCluster](#)
 - [Décrire les clusters](#)
 - [Décrire une déclaration](#)
 - [Exécuter une instruction](#)
 - [getStatementResult](#)
 - [listDatabasesPaginator](#)
 - [Modifier le cluster](#)

Exemples d'Amazon Rekognition utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Rekognition.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CompareFaces

L'exemple de code suivant montre comment utiliser `CompareFaces`.

Pour de plus amples informations, veuillez consulter [Comparaison de visages dans des images](#).

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <pathSource> <pathTarget>

                Where:
                pathSource - The path to the source image (for example, C:\\AWS\\
\\pic1.png).\\s
                pathTarget - The path to the target image (for example, C:\\AWS\\
\\pic2.png).\\s
    }
}
```

```
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    Float similarityThreshold = 70F;
    String sourceImage = args[0];
    String targetImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    compareTwoFaces(rekClient, similarityThreshold, sourceImage, targetImage);
    rekClient.close();
}

public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage,
    String targetImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        InputStream tarStream = new FileInputStream(targetImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        Image tarImage = Image.builder()
            .bytes(targetBytes)
            .build();

        CompareFacesRequest facesRequest = CompareFacesRequest.builder()
            .sourceImage(souImage)
            .targetImage(tarImage)
            .similarityThreshold(similarityThreshold)
            .build();

        // Compare the two images.
```

```
        CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
        List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
        for (CompareFacesMatch match : faceDetails) {
            ComparedFace face = match.face();
            BoundingBox position = face.boundingBox();
            System.out.println("Face at " + position.left().toString()
                + " " + position.top()
                + " matches with " + face.confidence().toString()
                + "% confidence.");

        }
        List<ComparedFace> uncomparing = compareFacesResult.unmatchedFaces();
        System.out.println("There was " + uncomparing.size() + " face(s) that did
not match");
        System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
        System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println("Failed to load source image " + sourceImage);
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CompareFaces](#) à la section Référence des AWS SDK for Java 2.x API.

CreateCollection

L'exemple de code suivant montre comment utiliser `CreateCollection`.

Pour plus d'informations, consultez [Création d'une collection](#).

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionName>\s

                Where:
                collectionName - The name of the collection.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```



```
        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void createMyCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
                .collectionId(collectionId)
                .build();

            CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
            System.out.println("Status code: " +
collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCollection](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteCollection

L'exemple de code suivant montre comment utiliser `DeleteCollection`.

Pour plus d'informations, consultez [Suppression d'une collection](#).

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>\s

                Where:
                collectionId - The id of the collection to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```

```
        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
                .collectionId(collectionId)
                .build();

            DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCollection](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteFaces

L'exemple de code suivant montre comment utiliser `DeleteFaces`.

Pour plus d'informations, veuillez consulter [Supprimer des visages d'une collection](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <faceId>\s

                Where:
                    collectionId - The id of the collection from which faces are
deleted.\s

                    faceId - The id of the face to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }
}
```

```
public static void deleteFacesCollection(RekognitionClient rekClient,
    String collectionId,
    String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteFaces](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeCollection

L'exemple de code suivant montre comment utiliser `DescribeCollection`.

Pour plus d'informations, veuillez consulter [Description d'une collection](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionName>

                Where:
                    collectionName - The name of the Amazon Rekognition collection.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                .collectionId(collectionName)
                .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
```

```
        .describeCollection(describeCollectionRequest);
        System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
        System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCollection](#) à la section Référence des AWS SDK for Java 2.x API.

DetectFaces

L'exemple de code suivant montre comment utiliser DetectFaces.

Pour plus d'informations, veuillez consulter [Détection des visages dans une image](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectFacesinImage(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectFacesinImage(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```



```
// Create an Image object for the source image.
Image souImage = Image.builder()
    .bytes(sourceBytes)
    .build();

DetectFacesRequest facesRequest = DetectFacesRequest.builder()
    .attributes(Attribute.ALL)
    .image(souImage)
    .build();

DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
List<FaceDetail> faceDetails = facesResponse.faceDetails();
for (FaceDetail face : faceDetails) {
    AgeRange ageRange = face.ageRange();
    System.out.println("The detected face is estimated to be between "
        + ageRange.low().toString() + " and " +
ageRange.high().toString()
        + " years old.");

    System.out.println("There is a smile : " +
face.smile().value().toString());
}

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectFaces](#) à la section Référence des AWS SDK for Java 2.x API.

DetectLabels

L'exemple de code suivant montre comment utiliser `DetectLabels`.

Pour plus d'informations, veuillez consulter [Détection des étiquettes dans une image](#).

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    detectImageLabels(rekClient, sourceImage);
    rekClient.close();
}

public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
            .image(souImage)
            .maxLabels(10)
            .build();

        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label : labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectLabels](#) à la section Référence des AWS SDK for Java 2.x API.

DetectModerationLabels

L'exemple de code suivant montre comment utiliser `DetectModerationLabels`.

Pour plus d'informations, veuillez consulter [Détection des images inappropriées](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectModLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                .image(souImage)
                .minConfidence(60F)
                .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                .detectModerationLabels(moderationLabelsRequest);
```

```
        List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");
        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name()
                + "\n Confidence: " + label.confidence().toString() + "%"
                + "\n Parent:" + label.parentName());
        }
    } catch (RekognitionException | FileNotFoundException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectModerationLabels](#) à la section Référence des AWS SDK for Java 2.x API.

DetectText

L'exemple de code suivant montre comment utiliser `DetectText`.

Pour plus d'informations, consultez [Détection de texte dans une image](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image that contains text (for
example, C:\\AWS\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectTextLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
Image souImage = Image.builder()
    .bytes(sourceBytes)
    .build();

DetectTextRequest textRequest = DetectTextRequest.builder()
    .image(souImage)
    .build();

DetectTextResponse textResponse = rekClient.detectText(textRequest);
List<TextDetection> textCollection = textResponse.textDetections();
System.out.println("Detected lines and words");
for (TextDetection text : textCollection) {
    System.out.println("Detected: " + text.detectedText());
    System.out.println("Confidence: " + text.confidence().toString());
    System.out.println("Id : " + text.id());
    System.out.println("Parent Id: " + text.parentId());
    System.out.println("Type: " + text.type());
    System.out.println();
}

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectText](#) à la section Référence des AWS SDK for Java 2.x API.

IndexFaces

L'exemple de code suivant montre comment utiliser `IndexFaces`.

Pour plus d'informations, veuillez consulter [Ajouter des visages à une collection](#).

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = ""

            Usage:      <collectionId> <sourceImage>

            Where:
                collectionName - The name of the collection.
```

```
        sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    addToCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        IndexFacesRequest facesRequest = IndexFacesRequest.builder()
            .collectionId(collectionId)
            .image(souImage)
            .maxFaces(1)
            .qualityFilter(QualityFilter.AUTO)
            .detectionAttributes(Attribute.DEFAULT)
            .build();

        IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
        System.out.println("Results for the image");
        System.out.println("\n Faces indexed:");
        List<FaceRecord> faceRecords = facesResponse.faceRecords();
        for (FaceRecord faceRecord : faceRecords) {
            System.out.println("  Face ID: " + faceRecord.face().faceId());
        }
    }
}
```

```
        System.out.println(" Location:" +
faceRecord.faceDetail().boundingBox().toString());
    }

    List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
    System.out.println("Faces not indexed:");
    for (UnindexedFace unindexedFace : unindexedFaces) {
        System.out.println(" Location:" +
unindexedFace.faceDetail().boundingBox().toString());
        System.out.println(" Reasons:");
        for (Reason reason : unindexedFace.reasons()) {
            System.out.println("Reason: " + reason);
        }
    }

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [IndexFaces](#) à la section Référence des AWS SDK for Java 2.x API.

ListCollections

L'exemple de code suivant montre comment utiliser `ListCollections`.

Pour en savoir plus, consultez [Répertoire de collections](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
                .maxResults(10)
                .build();

            ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }
        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [ListCollections](#) à la section Référence des AWS SDK for Java 2.x API.

ListFaces

L'exemple de code suivant montre comment utiliser `ListFaces`.

Pour plus d'informations, consultez [Répertoire de visages d'une collection](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>
```

```
        Where:
            collectionId - The name of the collection.\s
        """;

    if (args.length < 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Faces in collection " + collectionId);
    listFacesCollection(rekClient, collectionId);
    rekClient.close();
}

public static void listFacesCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        ListFacesRequest facesRequest = ListFacesRequest.builder()
            .collectionId(collectionId)
            .maxResults(10)
            .build();

        ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
        List<Face> faces = facesResponse.faces();
        for (Face face : faces) {
            System.out.println("Confidence level there is a face: " +
face.confidence());
            System.out.println("The face Id value is " + face.faceId());
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListFaces](#) à la section Référence des AWS SDK for Java 2.x API.

RecognizeCelebrities

L'exemple de code suivant montre comment utiliser `RecognizeCelebrities`.

Pour plus d'informations, consultez [Reconnaissance de célébrités dans une image](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
```

```
    final String usage = ""
        Usage:    <sourceImage>

        Where:
            sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Locating celebrities in " + sourceImage);
    recognizeAllCelebrities(rekClient, sourceImage);
    rekClient.close();
}

public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
            .image(souImage)
            .build();

        RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
        List<Celebrity> celebs = result.celebrityFaces();
        System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
        for (Celebrity celebrity : celebs) {
            System.out.println("Celebrity recognized: " + celebrity.name());
        }
    }
}
```



```
        System.out.println("Celebrity ID: " + celebrity.id());

        System.out.println("Further information (if available):");
        for (String url : celebrity.urls()) {
            System.out.println(url);
        }
        System.out.println();
    }
    System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [Reconnaitre des célébrités](#) à la section Référence des AWS SDK for Java 2.x API.

SearchFaces

L'exemple de code suivant montre comment utiliser `SearchFaces`.

Pour plus d'informations, veuillez consulter [Recherche d'un visage \(identification faciale\)](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
```

```
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }
}
```

```
}

    public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
            .maxFaces(10)
            .faceMatchThreshold(70F)
            .collectionId(collectionId)
            .build();

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [SearchFaces](#) à la section Référence des AWS SDK for Java 2.x API.

SearchFacesByImage

L'exemple de code suivant montre comment utiliser `SearchFacesByImage`.

Pour plus d'informations, voir [Recherche d'un visage \(image\)](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String collectionId = args[0];
String faceId = args[1];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Searching for a face in a collections");
searchFaceById(rekClient, collectionId, faceId);
rekClient.close();
}

public static void searchFaceById(RekognitionClient rekClient, String
collectionId, String faceId) {
    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [SearchFacesByImage](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Détecter les informations contenues dans les vidéos

L'exemple de code suivant illustre comment :

- Lancer des tâches sur Amazon Rekognition pour détecter des éléments tels que des personnes, des objets et du texte dans des vidéos.
- Vérifier l'état de la tâche jusqu'à ce qu'elle soit terminée.
- Afficher la liste des éléments détectés par chaque tâche.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez des informations sur des célébrités à partir d'une vidéo située dans un compartiment Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;
```

```
/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
    }
```

```
        .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();

        startCelebrityDetection(rekClient, channel, bucket, video);
        getCelebrityDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startCelebrityDetection(RekognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartCelebrityRecognitionRequest recognitionRequest =
                StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidObj)
                    .build();

            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
                rekClient
                    .startCelebrityRecognition(recognitionRequest);
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```



```
public static void getCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken = null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (recognitionResponse != null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
                recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
        }
    }
}
```

```

        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<CelebrityRecognition> celebs =
recognitionResponse.celebrities();
        for (CelebrityRecognition celeb : celebs) {
            long seconds = celeb.timestamp() / 1000;
            System.out.print("Sec: " + seconds + " ");
            CelebrityDetail details = celeb.celebrity();
            System.out.println("Name: " + details.name());
            System.out.println("Id: " + details.id());
            System.out.println();
        }

    } while (recognitionResponse.nextToken() != null);

} catch (RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}

```

Détectez les étiquettes dans une vidéo par une opération de détection d'étiquettes.

```

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;

```

```
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s
                queueUrl- The URL of a SQS queue.\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
```

```
String queueUrl = args[2];
String topicArn = args[3];
String roleArn = args[4];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

SqsClient sqs = SqsClient.builder()
    .region(Region.US_EAST_1)
    .build();

NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startLabels(rekClient, channel, bucket, video);
getLabelJob(rekClient, sqs, queueUrl);
System.out.println("This example is done!");
sqs.close();
rekClient.close();
}

public static void startLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidObj)
            .minConfidence(50F)
```

```
        .build());

        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy + " status is: " + status);

            Thread.sleep(1000);
            yy++;
        }

        System.out.println(startJobId + " status is: " + status);

    } catch (RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();
```

```
try {
    messages = sqs.receiveMessage(messageRequest).messages();

    if (!messages.isEmpty()) {
        for (Message message : messages) {
            String notification = message.body();

            // Get the status and job id from the notification
            ObjectMapper mapper = new ObjectMapper();
            JsonNode jsonMessageTree = mapper.readTree(notification);
            JsonNode messageBodyText = jsonMessageTree.get("Message");
            ObjectMapper operationResultMapper = new ObjectMapper();
            JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
            JsonNode operationJobId = jsonResultTree.get("JobId");
            JsonNode operationStatus = jsonResultTree.get("Status");
            System.out.println("Job found in JSON is " + operationJobId);

            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

            String jobId = operationJobId.textValue();
            if (startJobId.compareTo(jobId) == 0) {
                System.out.println("Job id: " + operationJobId);
                System.out.println("Status : " +
operationStatus.toString());

                if (operationStatus.asText().equals("SUCCEEDED"))
                    getResultsLabels(rekClient);
                else
                    System.out.println("Video analysis failed");

                sqs.deleteMessage(deleteMessageRequest);
            } else {
                System.out.println("Job received was not job " +
startJobId);

                sqs.deleteMessage(deleteMessageRequest);
            }
        }
    }
}
```

```
    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

    int maxResults = 10;
    String paginationToken = null;
    GetLabelDetectionResponse labelDetectionResult = null;

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels = labelDetectionResult.labels();
            for (LabelDetection detectedLabel : detectedLabels) {
                long seconds = detectedLabel.timestamp();
                Label label = detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");
            }
        } while (labelDetectionResult.nextToken() != null);
    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}
```

```

        System.out.println("    Label:" + label.name());
        System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

        List<Instance> instances = label.instances();
        System.out.println("    Instances of " + label.name());

        if (instances.isEmpty()) {
            System.out.println("        " + "None");
        } else {
            for (Instance instance : instances) {
                System.out.println("            Confidence: " +
instance.confidence().toString());
                System.out.println("            Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("    Parent labels for " + label.name() +
":");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("            " + parent.name());
            }
        }
        System.out.println();
    }
    } while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}
}
}

```

Détectez des visages dans une vidéo stockée dans un compartiment Amazon S3.


```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>
```

```
        Where:
            bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
            video - The name of the video (for example, people.mp4).\s
            queueUrl- The URL of a SQS queue.\s
            topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
            roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String queueUrl = args[2];
    String topicArn = args[3];
    String roleArn = args[4];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startLabels(rekClient, channel, bucket, video);
    getLabelJob(rekClient, sqs, queueUrl);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}

public static void startLabels(RekognitionClient rekClient,
    NotificationChannel channel,
```

```
        String bucket,
        String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
        StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidObj)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
        rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
            GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
            rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy + " status is: " + status);
        }
    }
}
```

```
        Thread.sleep(1000);
        yy++;
    }

    System.out.println(startJobId + " status is: " + status);

} catch (RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message : messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId) == 0) {
```

```
        System.out.println("Job id: " + operationJobId);
        System.out.println("Status : " +
operationStatus.toString());

        if (operationStatus.asText().equals("SUCCEEDED"))
            getResultsLabels(rekClient);
        else
            System.out.println("Video analysis failed");

        sqs.deleteMessage(deleteMessageRequest);
    } else {
        System.out.println("Job received was not job " +
startJobId);

        sqs.deleteMessage(deleteMessageRequest);
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

    int maxResults = 10;
    String paginationToken = null;
    GetLabelDetectionResponse labelDetectionResult = null;

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
```

```

        .maxResults(maxResults)
        .nextToken(paginationToken)
        .build();

    labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
    VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());

    List<LabelDetection> detectedLabels = labelDetectionResult.labels();
    for (LabelDetection detectedLabel : detectedLabels) {
        long seconds = detectedLabel.timestamp();
        Label label = detectedLabel.label();
        System.out.println("Millisecond: " + seconds + " ");

        System.out.println("    Label:" + label.name());
        System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

        List<Instance> instances = label.instances();
        System.out.println("    Instances of " + label.name());

        if (instances.isEmpty()) {
            System.out.println("        " + "None");
        } else {
            for (Instance instance : instances) {
                System.out.println("            Confidence: " +
instance.confidence().toString());
                System.out.println("            Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("    Parent labels for " + label.name() +
":");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("            " + parent.name());
            }
        }
    }
}

```

```
        }
    }
    System.out.println();
}
} while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
}
}
}
```

Détectez un contenu inapproprié ou offensant dans une vidéo stockée dans un compartiment Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```

```
*/
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }
}
```



```
public static void startModerationDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vidObj)
            .build();

        StartContentModerationResponse startModDetectionResult = rekClient
            .startContentModeration(modDetectionRequest);
        startJobId = startModDetectionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getModResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetContentModerationResponse modDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (modDetectionResponse != null)
                paginationToken = modDetectionResponse.nextToken();
```

```
        GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
        .jobId(startJobId)
        .nextToken(paginationToken)
        .maxResults(10)
        .build();

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse =
rekClient.getContentModeration(modRequest);
            status = modDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
        for (ContentModerationDetection mod : mods) {
            long seconds = mod.timestamp() / 1000;
            System.out.print("Mod label: " + seconds + " ");
            System.out.println(mod.moderationLabel().toString());
            System.out.println();
        }

        } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);
```

```
        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Détectez les segments de repères techniques et les segments de détection de prises de vue dans une vidéo stockée dans un compartiment Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        SqsClient sqs = SqsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();
    }
}
```

```
        startSegmentDetection(rekClient, channel, bucket, video);
        getSegmentResults(rekClient);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startSegmentDetection(RekognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = Video.builder()
                .s3Object(s3obj)
                .build();

            StartShotDetectionFilter cueDetectionFilter =
            StartShotDetectionFilter.builder()
                .minSegmentConfidence(60F)
                .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
            StartTechnicalCueDetectionFilter.builder()
                .minSegmentConfidence(60F)
                .build();

            StartSegmentDetectionFilters filters =
            StartSegmentDetectionFilters.builder()
                .shotFilter(cueDetectionFilter)
                .technicalCueFilter(technicalCueDetectionFilter)
                .build();

            StartSegmentDetectionRequest segDetectionRequest =
            StartSegmentDetectionRequest.builder()
                .jobTag("DetectingLabels")
                .notificationChannel(channel)
                .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                .video(vidObj)
```

```
        .filters(filters)
        .build();

        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                status = segDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
            }
        }
    }
}
```

```
        yy++;
    }
    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
    for (VideoMetadata metaData : videoMetaData) {
        System.out.println("Format: " + metaData.format());
        System.out.println("Codec: " + metaData.codec());
        System.out.println("Duration: " + metaData.durationMillis());
        System.out.println("FrameRate: " + metaData.frameRate());
        System.out.println("Job");
    }

    List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
    for (SegmentDetection detectedSegment : detectedSegments) {
        String type = detectedSegment.type().toString();
        if (type.contains(SegmentType.technical_cue.toString())) {
            System.out.println("Technical Cue");
            TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
            System.out.println("\tType: " + segmentCue.type());
            System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
        }

        if (type.contains(SegmentType.shot.toString())) {
            System.out.println("Shot");
            ShotSegment segmentShot = detectedSegment.shotSegment();
            System.out.println("\tIndex " + segmentShot.index());
            System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
        }

        long seconds = detectedSegment.durationMillis();
        System.out.println("\tDuration : " + seconds + " milliseconds");
        System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
        System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
        System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
    }
}
```

```
        System.out.println();
    }

    } while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Détectez le texte dans une vidéo stockée dans un compartiment Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""
```



```
Usage:    <bucket> <video> <topicArn> <roleArn>

Where:
    bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
    video - The name of video (for example, people.mp4).\s
    topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
    roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
    """;

if (args.length != 4) {
    System.out.println(usage);
    System.exit(1);
}

String bucket = args[0];
String video = args[1];
String topicArn = args[2];
String roleArn = args[3];

Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startTextLabels(rekClient, channel, bucket, video);
getTextResults(rekClient);
System.out.println("This example is done!");
rekClient.close();
}

public static void startTextLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
```

```
S3Object s3obj = S3Object.builder()
    .bucket(bucket)
    .name(video)
    .build();

Video vid0b = Video.builder()
    .s3object(s3obj)
    .build();

StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
    .jobTag("DetectingLabels")
    .notificationChannel(channel)
    .video(vid0b)
    .build();

StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
startJobId = labelDetectionResponse.jobId();

} catch (RekognitionException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}

public static void getTextResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetTextDetectionResponse textDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (textDetectionResponse != null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
    .jobId(startJobId)
    .nextToken(paginationToken)
    .maxResults(10)
    .build();
```

```
        // Wait until the job succeeds.
        while (!finished) {
            textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
            status = textDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
        for (TextDetectionResult detectedText : labels) {
            System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
            System.out.println("Id : " + detectedText.textDetection().id());
            System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
            System.out.println("Type: " +
detectedText.textDetection().type());
            System.out.println("Text: " +
detectedText.textDetection().detectedText());
            System.out.println();
        }

        } while (textDetectionResponse != null &&
textDetectionResponse.nextToken() != null);
```

```

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

Détectez des personnes dans une vidéo stockée dans un compartiment Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:

```

```

        bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
        video - The name of video (for example, people.mp4).\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startPersonLabels(rekClient, channel, bucket, video);
    getPersonDetectionResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startPersonLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();
    }
}

```

```
        Video vid0b = Video.builder()
            .s3object(s3obj)
            .build();

        StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
            .jobTag("DetectingLabels")
            .video(vid0b)
            .notificationChannel(channel)
            .build();

        StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getPersonDetectionResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetPersonTrackingResponse personTrackingResult = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (personTrackingResult != null)
                paginationToken = personTrackingResult.nextToken();

            GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
```

```
        personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
        status = personTrackingResult.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<PersonDetection> detectedPersons =
personTrackingResult.persons();
    for (PersonDetection detectedPerson : detectedPersons) {
        long seconds = detectedPerson.timestamp() / 1000;
        System.out.print("Sec: " + seconds + " ");
        System.out.println("Person Identifier: " +
detectedPerson.person().index());
        System.out.println();
    }

    } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [GetCelebrityRecognition](#)
 - [GetContentModeration](#)
 - [GetLabelDetection](#)
 - [GetPersonTracking](#)
 - [GetSegmentDetection](#)
 - [GetTextDetection](#)
 - [StartCelebrityRecognition](#)
 - [StartContentModeration](#)
 - [StartLabelDetection](#)
 - [StartPersonTracking](#)
 - [StartSegmentDetection](#)
 - [StartTextDetection](#)

Exemples d'enregistrement de domaine Route 53 à l'aide du SDK for Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de l'enregistrement de domaine AWS SDK for Java 2.x avec Route 53.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Enregistrement de domaine Route 53

Les exemples de code suivants montrent comment démarrer avec l'enregistrement de domaine Route 53.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code examples performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the "com" type
 * and displays the prices for Registration and Renewal.
 */
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <hostedZoneId> \n\n" +
```

```

        "Where:\n" +
        "    hostedZoneId - The id value of an existing hosted zone. \n";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String domainType = args[0];
    Region region = Region.US_EAST_1;
    Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Invokes ListPrices for at least one domain type.");
    listPrices(route53DomainsClient, domainType);
    System.out.println(DASHES);
}

public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .maxItems(10)
            .tld(domainType)
            .build();

        ListPricesResponse response =
route53DomainsClient.listPrices(pricesRequest);
        List<DomainPrice> prices = response.prices();
        for (DomainPrice pr : prices) {
            System.out.println("Name: " + pr.name());
            System.out.println(
                "Registration: " + pr.registrationPrice().price() + " " +
pr.registrationPrice().currency());
            System.out.println("Renewal: " + pr.renewalPrice().price() + " " +
pr.renewalPrice().currency());
            System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
            System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
            System.out.println("Change Ownership: " +
pr.changeOwnershipPrice().price() + " "

```

```
        + pr.changeOwnershipPrice().currency());
        System.out.println(
            "Restoration: " + pr.restorationPrice().price() + " " +
pr.restorationPrice().currency());
        System.out.println(" ");
    }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListPrices](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CheckDomainAvailability

L'exemple de code suivant montre comment utiliser `CheckDomainAvailability`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
```

```
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
        .domainName(domainSuggestion)
        .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
        .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CheckDomainAvailability](#) à la section Référence des AWS SDK for Java 2.x API.

CheckDomainTransferability

L'exemple de code suivant montre comment utiliser `CheckDomainTransferability`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
        .domainName(domainSuggestion)
        .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
```

```
        .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CheckDomainTransferability](#) à la section Référence des AWS SDK for Java 2.x API.

GetDomainDetail

L'exemple de code suivant montre comment utiliser `GetDomainDetail`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
    try {
        GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
            .domainName(domainSuggestion)
            .build();

        GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
        System.out.println("The contact first name is " +
response.registrantContact().firstName());
        System.out.println("The contact last name is " +
response.registrantContact().lastName());
        System.out.println("The contact org name is " +
response.registrantContact().organizationName());
    }
```

```
    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetDomainDetail](#) à la section Référence des AWS SDK for Java 2.x API.

GetDomainSuggestions

L'exemple de code suivant montre comment utiliser `GetDomainSuggestions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
            .domainName(domainSuggestion)
            .suggestionCount(5)
            .onlyAvailable(true)
            .build();

        GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
        List<DomainSuggestion> suggestions = response.suggestionsList();
        for (DomainSuggestion suggestion : suggestions) {
            System.out.println("Suggestion Name: " + suggestion.domainName());
            System.out.println("Availability: " + suggestion.availability());
            System.out.println(" ");
        }
    }
}
```

```
    } catch (Route53Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetDomainSuggestions](#) à la section Référence des AWS SDK for Java 2.x API.

GetOperationDetail

L'exemple de code suivant montre comment utiliser `GetOperationDetail`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getOperationalDetail(Route53DomainsClient  
route53DomainsClient, String operationId) {  
    try {  
        GetOperationDetailRequest detailRequest =  
        GetOperationDetailRequest.builder()  
            .operationId(operationId)  
            .build();  
  
        GetOperationDetailResponse response =  
        route53DomainsClient.getOperationDetail(detailRequest);  
        System.out.println("Operation detail message is " + response.message());  
  
    } catch (Route53Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetOperationDetail](#) à la section Référence des AWS SDK for Java 2.x API.

ListDomains

L'exemple de code suivant montre comment utiliser `ListDomains`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
            .flatMap(r -> r.domains().stream())
            .forEach(content -> System.out.println("The domain name is " +
content.domainName()));
    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDomains](#) à la section Référence des AWS SDK for Java 2.x API.

ListOperations

L'exemple de code suivant montre comment utiliser `ListOperations`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
            .submittedSince(myTime)
            .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
            .flatMap(r -> r.operations().stream())
            .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListOperations](#) à la section Référence des AWS SDK for Java 2.x API.

ListPrices

L'exemple de code suivant montre comment utiliser `ListPrices`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
            .build();

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListPrices](#) à la section Référence des AWS SDK for Java 2.x API.

RegisterDomain

L'exemple de code suivant montre comment utiliser `RegisterDomain`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
    String domainSuggestion,
    String phoneNumber,
    String email,
    String firstName,
    String lastName,
    String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
```

```
        .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [RegisterDomain](#) à la section Référence des AWS SDK for Java 2.x API.

ViewBilling

L'exemple de code suivant montre comment utiliser `ViewBilling`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);
```

```
ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
    .start(myStartTime)
    .end(myEndTime)
    .build();

ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
listRes.stream()
    .flatMap(r -> r.billingRecords().stream())
    .forEach(content -> System.out.println(" Bill Date: " +
content.billDate() +
        " Operation: " + content.operationAsString() +
        " Price: " + content.price()));

} catch (Route53Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ViewBilling](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Commencer avec les domaines

L'exemple de code suivant illustre comment :

- Répertorier les domaines actuels et les opérations effectuées au cours de l'année écoulée.
- Afficher la facturation de l'année écoulée et les prix des types de domaines.
- Obtenir des suggestions de domaines.
- Vérifier la disponibilité et la transférabilité du domaine.
- Éventuellement, demander l'enregistrement d'un domaine.
- Obtenir des informations sur une opération.
- Éventuellement, obtenir des informations sur un domaine.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <domainType> <phoneNumber> <email> <domainSuggestion>
<firstName> <lastName> <city>

Where:
    domainType - The domain type (for example, com).\s
    phoneNumber - The phone number to use (for example,
+91.9966564xxx)    email - The email address to use.    domainSuggestion - The
domain suggestion (for example, findmy.accountants).\s
    firstName - The first name to use to register a domain.\s
    lastName - The last name to use to register a domain.\s
    city - the city to use to register a domain.\s
    """;

if (args.length != 7) {
    System.out.println(usage);
    System.exit(1);
}

String domainType = args[0];
String phoneNumber = args[1];
String email = args[2];
String domainSuggestion = args[3];
String firstName = args[4];
String lastName = args[5];
String city = args[6];
Region region = Region.US_EAST_1;
Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. List current domains.");
listDomains(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List operations in the past year.");

```

```
listOperations(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. View billing for the account in the past year.");
listBillingRecords(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. View prices for domain types.");
listPrices(route53DomainsClient, domainType);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get domain suggestions.");
listDomainSuggestions(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Check domain availability.");
checkDomainAvailability(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Check domain transferability.");
checkDomainTransferability(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Request a domain registration.");
String opId = requestDomainRegistration(route53DomainsClient,
    domainSuggestion, phoneNumber, email, firstName,
    lastName, city);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get operation details.");
getOperationalDetail(route53DomainsClient, opId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get domain details.");
System.out.println("Note: You must have a registered domain to get
details.");
```



```
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxxx not found in xxxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                .domainName(domainSuggestion)
                .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                .operationId(operationId)
                .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
        String domainSuggestion,
        String phoneNumber,
        String email,
        String firstName,
        String lastName,
        String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
            .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
```

```

    try {
        GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
            .domainName(domainSuggestion)
            .suggestionCount(5)
            .onlyAvailable(true)
            .build();

        GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
        List<DomainSuggestion> suggestions = response.suggestionsList();
        for (DomainSuggestion suggestion : suggestions) {
            System.out.println("Suggestion Name: " + suggestion.domainName());
            System.out.println("Availability: " + suggestion.availability());
            System.out.println(" ");
        }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
            .build();

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
    }
}

```

```
        System.exit(1);
    }
}

public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);

        ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
            .start(myStartTime)
            .end(myEndTime)
            .build();

        ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
        listRes.stream()
            .flatMap(r -> r.billingRecords().stream())
            .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                " Operation: " + content.operationAsString() +
                " Price: " + content.price()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);
```

```
        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
        .submittedSince(myTime)
        .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
        .flatMap(r -> r.operations().stream())
        .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
        .flatMap(r -> r.domains().stream())
        .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CheckDomainAvailability](#)
 - [CheckDomainTransferability](#)
 - [GetDomainDetail](#)

- [GetDomainSuggestions](#)
- [GetOperationDetail](#)
- [ListDomains](#)
- [ListOperations](#)
- [ListPrices](#)
- [RegisterDomain](#)
- [ViewBilling](#)

Exemples d'Amazon S3 utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon S3.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Hello Amazon S3

Les exemples de code suivants montrent comment démarrer avec Amazon S3.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)
- [Exemples sans serveur](#)

Actions

CopyObject

L'exemple de code suivant montre comment utiliser `CopyObject`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Copiez un objet en utilisant un [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```
        <objectKey> <fromBucket> <toBucket>

    Where:
        objectKey - The name of the object (for example, book.pdf).
        fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
        toBucket - The S3 bucket to copy the object to (for example,
bucket2).

        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String objectKey = args[0];
    String fromBucket = args[1];
    String toBucket = args[2];
    System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    copyBucketObject(s3, fromBucket, objectKey, toBucket);
    s3.close();
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}
}
```

Utilisez un [S3 TransferManager](#) pour [copier un objet d'un compartiment vers un autre](#). Consultez le [fichier complet](#) et le [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK for Java 2.x API.

CreateBucket

L'exemple de code suivant montre comment utiliser `CreateBucket`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez un compartiment.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the bucket to create. The bucket name
            must be unique, or an error occurs.
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Creating a bucket named %s\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    createBucket(s3, bucketName);
    s3.close();
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Créez un compartiment avec le verrouillage des objets activé.

```
// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteBucket

L'exemple de code suivant montre comment utiliser `DeleteBucket`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();
```

```
s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteBucketPolicy

L'exemple de code suivant montre comment utiliser DeleteBucketPolicy.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>
```

```
        Where:
            bucketName - The Amazon S3 bucket to delete the policy from (for
example, bucket1).""";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteS3BucketPolicy(s3, bucketName);
    s3.close();
}

// Delete the bucket policy.
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteBucketWebsite

L'exemple de code suivant montre comment utiliser `DeleteBucketWebsite`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketWebsite](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteObjects

L'exemple de code suivant montre comment utiliser `DeleteObjects`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName>

                Where:
                    bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String bucketName = args[0];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

deleteBucketObjects(s3, bucketName);
s3.close();
}

public static void deleteBucketObjects(S3Client s3, String bucketName) {
    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(del)
            .build();
```

```
s3.deleteObjects(multiObjectDeleteRequest);
System.out.println("Multiple objects are deleted!");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK for Java 2.x API.

GetBucketAcl

L'exemple de code suivant montre comment utiliser `GetBucketAcl`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class GetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <objectKey>

            Where:
                bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
                objectKey - The object to get the ACL for.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);

```

```
        List<Grant> grants = aclRes.grants();
        String grantee = "";
        for (Grant grant : grants) {
            System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
            grantee = grant.grantee().id();
        }

        return grantee;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section Référence des AWS SDK for Java 2.x API.

GetBucketPolicy

L'exemple de code suivant montre comment utiliser `GetBucketPolicy`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
```

```
public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
```



```
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

GetObject

L'exemple de code suivant montre comment utiliser `GetObject`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Lisez des données sous forme de tableau d'octets en utilisant un [S3Client](#).

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
    keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
```

```
        .bucket(bucketName)
        .build();

    ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
    byte[] data = objectBytes.asByteArray();

    // Write the data to a local file.
    File myFile = new File(path);
    OutputStream os = new FileOutputStream(myFile);
    os.write(data);
    System.out.println("Successfully obtained bytes from an S3 object");
    os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Utilisez un [S3 TransferManager](#) pour [télécharger un objet](#) d'un compartiment S3 vers un fichier local. Consultez le [fichier complet](#) et le [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```

import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                            String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentType());
        return downloadResult.response().contentType();
    }

```

Lisez les étiquettes qui appartiennent à un objet à l'aide d'un [S3Client](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

```

```
Usage:
    <bucketName> <keyName>\s

Where:
    bucketName - The Amazon S3 bucket name.\s
    keyName - A key name that represents the object.\s
""";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

listTags(s3, bucketName, keyName);
s3.close();
}

public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

Obtenez une URL pour un objet en utilisant un [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>\s

                Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
        getURL(s3, bucketName, keyName);
        s3.close();
    }

    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Obtenez un objet en utilisant l'objet client S3Presigner via un [S3Client](#).

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();
```



```

        PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
        String theUrl = presignedGetObjectRequest.url().toString();
        System.out.println("Presigned URL: " + theUrl);
        HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
        presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
            values.forEach(value -> {
                connection.addRequestProperty(header, value);
            });
        });

        // Send any request payload that the service needs (not needed when
// isBrowserExecutable is true).
        if (presignedGetObjectRequest.signedPayload().isPresent()) {
            connection.setDoOutput(true);

            try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                OutputStream httpOutputStream =
connection.getOutputStream()) {
                IoUtils.copy(signedPayload, httpOutputStream);
            }
        }

        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            System.out.println("Service returned response: ");
            IoUtils.copy(content, System.out);
        }

    } catch (S3Exception | IOException e) {
        e.printStackTrace();
    }
}
}

```

Obtenez un objet en utilisant un `ResponseTransformer` objet et [S3Client](#).

```
import software.amazon.awssdk.core.ResponseBytes;
```

```
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```
        .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK for Java 2.x API.

GetObjectLegalHold

L'exemple de code suivant montre comment utiliser `GetObjectLegalHold`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "'");
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLegalHold](#) à la section Référence des AWS SDK for Java 2.x API.

GetObjectLockConfiguration

L'exemple de code suivant montre comment utiliser `GetObjectLockConfiguration`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .build();


    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().getObjectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section Référence des AWS SDK for Java 2.x API.

GetObjectRetention

L'exemple de code suivant montre comment utiliser `GetObjectRetention`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key +" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+ ".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectRetention](#) à la section Référence des AWS SDK for Java 2.x API.

HeadObject

L'exemple de code suivant montre comment utiliser `HeadObject`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Déterminez le type de contenu d'un objet.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>>

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
    keyName) {
        try {
```

```

        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        HeadObjectResponse objectHead = s3.headObject(objectRequest);
        String type = objectHead.contentType();
        System.out.println("The object content type is " + type);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

Obtenez le statut de restauration d'un objet.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];

```



```
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

checkStatus(s3, bucketName, keyName);
s3.close();
}

public static void checkStatus(S3Client s3, String bucketName, String keyName) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        HeadObjectResponse response = s3.headObject(headObjectRequest);
        System.out.println("The Amazon S3 object restoration status is " +
response.restore());

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [HeadObject](#) à la section Référence des AWS SDK for Java 2.x API.

ListBuckets

L'exemple de code suivant montre comment utiliser `ListBuckets`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK for Java 2.x API.

ListMultipartUploads

L'exemple de code suivant montre comment utiliser `ListMultipartUploads`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The name of the Amazon S3 bucket where an in-
                progress multipart upload is occurring.
```

```
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();
    listUploads(s3, bucketName);
    s3.close();
}

public static void listUploads(S3Client s3, String bucketName) {
    try {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListMultipartUploads](#) à la section Référence des AWS SDK for Java 2.x API.

ListObjectsV2

L'exemple de code suivant montre comment utiliser ListObjectsV2.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.println("\n The name of the key is " + myValue.key());
            System.out.println("\n The object is " + calKb(myValue.size()) + "
KBs");
            System.out.println("\n The owner is " + myValue.owner());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calKb(Long val) {
    return val / 1024;
}
}
```

Lister les objets en utilisant la pagination.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .maxKeys(1)
                    .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                    .flatMap(r -> r.contents().stream())

```

```
        .forEach(content -> System.out.println(" Key: " + content.key()
+ " size = " + content.size())));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK for Java 2.x API.

PutBucketAcl

L'exemple de code suivant montre comment utiliser `PutBucketAcl`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <id>\s

            Where:
                bucketName - The Amazon S3 bucket to grant permissions on.\s
                id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)
                    .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();
```

```
List<Grant> grantList2 = new ArrayList<>();
grantList2.add(ownerGrant);

AccessControlPolicy acl = AccessControlPolicy.builder()
    .owner(builder -> builder.id(id))
    .grants(grantList2)
    .build();

PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
    .bucket(bucketName)
    .accessControlPolicy(acl)
    .build();

s3.putBucketAcl(putAclReq);

} catch (S3Exception e) {
    e.printStackTrace();
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAcl](#) à la section Référence des AWS SDK for Java 2.x API.

PutBucketCors

L'exemple de code suivant montre comment utiliser `PutBucketCors`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <accountId>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3
bucket.

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
    }
}
```

```
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketCors](#) à la section Référence des AWS SDK for Java 2.x API.

PutBucketLifecycleConfiguration

L'exemple de code suivant montre comment utiliser `PutBucketLifecycleConfiguration`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <bucketName> <accountId>\s

Where:
    bucketName - The Amazon Simple Storage Service
(Amazon S3) bucket to upload an object into.
    accountId - The id of the account that owns the
Amazon S3 bucket.

    """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String accountId = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setLifecycleConfig(s3, bucketName, accountId);
    getLifecycleConfig(s3, bucketName, accountId);
    deleteLifecycleConfig(s3, bucketName, accountId);
    System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
    s3.close();
}

    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
            // S3 Glacier.
            LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()

```

```
.storageClass(TransitionStorageClass.GLACIER)
    .days(0)
    .build();

LifecycleRule rule1 = LifecycleRule.builder()
    .id("Archive immediately rule")
    .filter(ruleFilter)
    .transitions(transition)
    .status(ExpirationStatus.ENABLED)
    .build();

// Create a second rule.
Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
    .days(0)
    .build();

List<Transition> transitionList = new ArrayList<>();
transitionList.add(transition2);

LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
    .prefix("glacierobjects/")
    .build();

LifecycleRule rule2 = LifecycleRule.builder()
    .id("Archive and then delete rule")
    .filter(ruleFilter2)
    .transitions(transitionList)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the LifecycleRule objects to an ArrayList.
ArrayList<LifecycleRule> ruleList = new ArrayList<>();
ruleList.add(rule1);
ruleList.add(rule2);

BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(ruleList)
    .build();
```



```
        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
        .builder()
        .bucket(bucketName)

        .lifecycleConfiguration(lifecycleConfiguration)
        .expectedBucketOwner(accountId)
        .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Retrieve the configuration and add a new rule.
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
            .builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag
predicate.

            LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();
```

```
        Transition transition = Transition.builder()

        .storageClass(TransitionStorageClass.GLACIER)
            .days(3650)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("NewRule")
            .filter(ruleFilter)
            .transitions(transition)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Add the new rule to the list.
        newList.add(rule1);
        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
            .rules(newList)
            .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
            .builder()
            .bucket(bucketName)

        .lifecycleConfiguration(lifecycleConfiguration)
            .expectedBucketOwner(accountId)
            .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the configuration from the Amazon S3 bucket.
    public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
        try {
```

```

        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [PutBucketLifecycleConfiguration](#) à la section Référence des AWS SDK for Java 2.x API.

PutBucketNotificationConfiguration

L'exemple de code suivant montre comment utiliser `PutBucketNotificationConfiguration`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;

```

```
import java.util.List;

public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket.\s
                topicArn - The Simple Notification Service topic ARN.\s
                id - An id value used for the topic configuration. This value is
displayed in the AWS Management Console.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String topicArn = args[1];
        String id = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3Client = S3Client.builder()
            .region(region)
            .build();

        setBucketNotification(s3Client, bucketName, topicArn, id);
        s3Client.close();
    }

    public static void setBucketNotification(S3Client s3Client, String bucketName,
String topicArn, String id) {
        try {
            List<Event> events = new ArrayList<>();
            events.add(Event.S3_OBJECT_CREATED_PUT);

            TopicConfiguration config = TopicConfiguration.builder()
                .topicArn(topicArn)
                .events(events)
                .id(id)
                .build();
```

```
List<TopicConfiguration> topics = new ArrayList<>();
topics.add(config);

NotificationConfiguration configuration =
NotificationConfiguration.builder()
    .topicConfigurations(topics)
    .build();

PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
    .builder()
    .bucket(bucketName)
    .notificationConfiguration(configuration)
    .skipDestinationValidation(true)
    .build();

// Set the bucket notification configuration.
s3Client.putBucketNotificationConfiguration(configurationRequest);
System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketNotificationConfiguration](#) à la section Référence des AWS SDK for Java 2.x API.

PutBucketPolicy

L'exemple de code suivant montre comment utiliser `PutBucketPolicy`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <polFile>

                Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
                Readme for an example).\s
                """;

        if (args.length != 2) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String polFile = args[1];
    String policyText = getBucketPolicyFromFile(polFile);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setPolicy(s3, bucketName, policyText);
    s3.close();
}

public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();

        s3.putBucketPolicy(policyReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}

// Loads a JSON-formatted policy from a file
public static String getBucketPolicyFromFile(String policyFile) {

    StringBuilder fileText = new StringBuilder();
```

```
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

PutBucketWebsite

L'exemple de code suivant montre comment utiliser PutBucketWebsite.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucketName> [indexdoc]\s

            Where:
                bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc    - The index document, ex. 'index.html'
                            If not specified, 'index.html' will be set.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }
}
```

```
    }

    public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                .build();

            PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .websiteConfiguration(websiteConfig)
                .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des AWS SDK for Java 2.x API.

PutObject

L'exemple de code suivant montre comment utiliser `PutObject`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Chargez un fichier dans un compartiment à l'aide d'un [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <objectKey> <objectPath>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```
        .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Utilisez un [S3 TransferManager](#) pour [télécharger un fichier](#) dans un compartiment. Consultez le [fichier complet](#) et le [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
```

```
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

public String uploadFile(S3TransferManager transferManager, String bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Chargez un objet dans un compartiment et définissez des étiquettes à l'aide d'un [S3Client](#).

```
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Tag tag1 = Tag.builder()
            .key("Tag 1")
            .value("This is tag 1")
            .build();

        Tag tag2 = Tag.builder()
            .key("Tag 2")
            .value("This is tag 2")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag1);
        tags.add(tag2);

        Tagging allTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
```

```
        .key(objectKey)
        .tagging(allTags)
        .build();

    s3.putObject(putObj, RequestBody.fromBytes(getObjectFile(objectPath)));

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag : obTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

        // Replace the object's tags with two new tags.
        Tag tag3 = Tag.builder()
            .key("Tag 3")
            .value("This is tag 3")
            .build();

        Tag tag4 = Tag.builder()
            .key("Tag 4")
            .value("This is tag 4")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag3);
        tags.add(tag4);
    }
}
```

```
        Tagging updatedTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(updatedTags)
            .build();

        s3.putObjectTagging(taggingRequest1);
        GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
        List<Tag> modTags = getTaggingRes2.tagSet();
        for (Tag sinTag : modTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
```

```

        e.printStackTrace();
    }
}
}

return byteArray;
}
}

```

Chargez un objet dans un compartiment et définissez les métadonnées à l'aide d'un [S3Client](#).

```

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

                Usage:
                <bucketName> <objectKey> <objectPath>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

```



```
        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

Chargez un objet dans un compartiment et définissez une valeur de conservation de l'objet à l'aide d'un [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <key> <bucketName>\s

                Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
                (for example, bucket1).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String key = args[0];
String bucketName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setRetentionPeriod(s3, key, bucketName);
s3.close();
}

public static void setRetentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();

        // To set Retention on an object, the Amazon S3 bucket must support
object
        // locking, otherwise an exception is thrown.
s3.putObjectRetention(retentionRequest);
        System.out.print("An object retention configuration was successfully
placed on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK for Java 2.x API.

PutObjectLegalHold

L'exemple de code suivant montre comment utiliser `PutObjectLegalHold`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLegalHold](#) à la section Référence des AWS SDK for Java 2.x API.

PutObjectLockConfiguration

L'exemple de code suivant montre comment utiliser `PutObjectLockConfiguration`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Définissez la configuration du verrouillage des objets d'un bucket.

```
// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
            .build();
```

```
        .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on "+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage() +
        "'");
    }
}
```

Définissez la période de rétention par défaut d'un bucket.

```
// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention retention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(retention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
        .build();
}
```

```
PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
System.out.println("Added a default retention to bucket "+bucketName +".");
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLockConfiguration](#) à la section Référence des AWS SDK for Java 2.x API.

PutObjectRetention

L'exemple de code suivant montre comment utiliser `PutObjectRetention`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time zone.
    ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);
}
```

```
// Print the formatted date string.
System.out.println("Formatted Date: " + humanReadableDate);
ObjectLockRetention retention = ObjectLockRetention.builder()
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .retainUntilDate(futureInstant)
    .build();

PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

getClient().putObjectRetention(retentionRequest);
System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section Référence des AWS SDK for Java 2.x API.

RestoreObject

L'exemple de code suivant montre comment utiliser `RestoreObject`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
```



```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 * For more information about restoring an object, see "Restoring an archived
 object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName> <expectedBucketOwner>

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value
of Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
    }
}
```

```
s3.close();
}

public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [RestoreObject](#) à la section Référence des AWS SDK for Java 2.x API.

SelectObjectContent

L'exemple de code suivant montre comment utiliser `SelectObjectContent`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

L'exemple suivant montre une requête utilisant un objet JSON. L'[exemple complet](#) montre également l'utilisation d'un objet CSV.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

```
public class SelectObjectContentExample {
    static final Logger logger =
    LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
    master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
    master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
        SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where c.area
    < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
            .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
            .compressionType(CompressionType.NONE)
            .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
            .json(JSONOutput.builder().recordDelimiter(null).build())
            .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
```

```

        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
    // Call the selectObjectContent method with the request and a response
    handler.
    // Supply an EventStreamInfo object to the response handler to gather
    records and information from the response.
    s3AsyncClient.selectObjectContent(select,
    buildResponseHandler(eventStreamInfo)).join();

    // Log out information gathered while processing the response stream.
    long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
        record.split("\n").length
    ).sum();
    logger.info("Total records {}: {}", fileType, recordCount);
    logger.info("Visitor onRecords for fileType {} called {} times", fileType,
    eventStreamInfo.getCountOnRecordsCalled());
    logger.info("Visitor onStats for fileType {}, {}", fileType,
    eventStreamInfo.getStats());
    logger.info("Visitor onContinuations for fileType {}, {}", fileType,
    eventStreamInfo.getCountContinuationEvents());
    return eventStreamInfo;
}

static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
    // Use a Visitor to process the response stream. This visitor logs
    information and gathers details while processing.
    final SelectObjectContentResponseHandler.Visitor visitor =
    SelectObjectContentResponseHandler.Visitor.builder()
        .onRecords(r -> {
            logger.info("Record event received.");
            eventStreamInfo.addRecord(r.payload().asUtf8String());
            eventStreamInfo.incrementOnRecordsCalled();
        })
        .onCont(ce -> {
            logger.info("Continuation event received.");
            eventStreamInfo.incrementContinuationEvents();
        })
}

```

```

        })
        .onProgress(pe -> {
            Progress progress = pe.details();
            logger.info("Progress event received:\n bytesScanned:
{} \n bytesProcessed: {} \n bytesReturned: {}",
                progress.bytesScanned(),
                progress.bytesProcessed(),
                progress.bytesReturned());
        })
        .onEnd(ee -> logger.info("End event received. "))
        .onStats(se -> {
            logger.info("Stats event received.");
            eventStreamInfo.addStats(se.details());
        })
        .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
    // processes the stream.
    return SelectObjectContentResponseHandler.builder()
        .subscriber(visitor).build();
}

// The EventStreamInfo class is used to store information gathered while
// processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }
}

```

```
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }

    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SelectObjectContent](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Créer une URL présignée

L'exemple de code suivant montre comment créer une URL présignée pour Amazon S3 et télécharger un objet.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Générez une URL pré-signée pour un objet, puis téléchargez-le (requête GET).

Importations.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Générez l'URL.

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();
```



```

        GetObjectPresignRequest presignRequest =
GetObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(10)) // The URL will
expire in 10 minutes.
        .getObjectRequest(objectRequest)
        .build();

        PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Téléchargez l'objet en utilisant l'une des trois approches suivantes.

Utilisez la classe JDK `URLConnection` (depuis v1.1) pour effectuer le téléchargement.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Utilisez la classe JDK `HttpClient` (depuis la version 11) pour effectuer le téléchargement.

```
/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());
    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

Utilisez la classe AWS SDK for Java `SdkHttpClient` pour effectuer le téléchargement.

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();
```

```

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Générez une URL pré-signée pour un téléchargement, puis chargez un fichier (requête PUT).

Importations.

```

import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;

```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Générez l'URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires
in 10 minutes.
            .putObjectRequest(objectRequest)
            .build();
```

```

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignedRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Téléchargez un objet de fichier en utilisant l'une des trois approches suivantes.

Utilisez la classe JDK `URLConnection` (depuis la version 1.1) pour effectuer le téléchargement.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));

        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        }
    }
}

```

```

        }
    } catch (IOException e) {
        logger.error(e.getMessage(), e);
    }

    out.close();
    connection.getResponseCode();
    logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Utilisez la classe JDK `HttpClient` (depuis la version 11) pour effectuer le téléchargement.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())

        .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Utilisez la `SdkHttpClient` classe AWS for Java V2 pour effectuer le téléchargement.

```
/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k,
v));

        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

Supprimer les téléchargements partitionnés incomplets

L'exemple de code suivant montre comment supprimer ou arrêter les téléchargements partitionnés incomplets sur Amazon S3.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Pour arrêter les téléchargements partitionnés en cours ou incomplets pour une quelconque raison, vous pouvez obtenir une liste des téléchargements, puis les supprimer comme indiqué dans l'exemple suivant.

```
public static void abortIncompleteMultipartUploadsFromList() {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(upload.key())
            .expectedBucketOwner(accountId)
            .uploadId(upload.uploadId())
            .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
        }
    }
}
```


Pour supprimer des téléchargements partitionnés incomplets initiés avant ou après une date, vous pouvez supprimer les téléchargements partitionnés de manière sélective en fonction d'un moment donné, comme indiqué dans l'exemple suivant.

```

static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
    .bucket(bucketName)
    .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
        if (upload.initiated().isBefore(pointInTime)) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
            }
        }
    }
}
}

```

Si vous avez accès à l'identifiant de téléchargement après avoir commencé un téléchargement en plusieurs parties, vous pouvez supprimer le téléchargement en cours à l'aide de cet identifiant.

```

static void abortMultipartUploadUsingUploadId() {
    String uploadId = startUploadReturningUploadId();
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b

```

```

        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

```

Pour supprimer systématiquement les téléchargements partitionnés incomplets datant de plus d'un certain nombre de jours, configurez une configuration du cycle de vie du bucket pour le bucket. L'exemple suivant montre comment créer une règle pour supprimer les téléchargements incomplets datant de plus de 7 jours.

```

static void abortMultipartUploadsUsingLifecycleConfig() {
    Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    // If the action is successful, the service sends back an HTTP 200 response
    with an empty HTTP body.
    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}
}

```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

Télécharger des objets dans un répertoire local

L'exemple de code suivant montre comment télécharger tous les objets d'un compartiment Amazon Simple Storage Service (Amazon S3) dans un répertoire local.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez un [S3 TransferManager](#) pour [télécharger tous les objets S3](#) dans le même compartiment S3. Consultez le [fichier complet](#) et le [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;
```

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

- Pour plus de détails sur l'API, reportez-vous [DownloadDirectory](#) à la section Référence des AWS SDK for Java 2.x API.

Démarrer avec les compartiments et les objets

L'exemple de code suivant illustre comment :

- créer un compartiment et y charger un fichier ;
- télécharger un objet à partir d'un compartiment ;
- copier un objet dans le sous-dossier d'un compartiment ;
- répertorier les objets d'un compartiment ;
- supprimer le compartiment et tous les objets qui y figurent.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Deletes the object from the Amazon S3 bucket.
 * 8. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The Amazon S3 bucket to create.
                key - The key to use.
                objectPath - The path where the file is located (for example,
                C:/AWS/book2.pdf).
                savePath - The path where the file is saved after it's
                downloaded (for example, C:/AWS/book2.pdf).
                toBucket - An Amazon S3 bucket to where an object is copied to
                (for example, C:/AWS/book2.pdf).\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
}

String bucketName = args[0];
String key = args[1];
String objectPath = args[2];
String savePath = args[3];
String toBucket = args[4];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon S3 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an Amazon S3 bucket.");
createBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Update a local file to the Amazon S3 bucket.");
uploadLocalFile(s3, bucketName, key, objectPath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Download the object to another local file.");
getObjectBytes(s3, bucketName, key, savePath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Perform a multipart upload.");
String multipartKey = "multiPartKey";
multipartUpload(s3, toBucket, multipartKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List all objects located in the Amazon S3 bucket.");
listAllObjects(s3, bucketName);
anotherListExample(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
        System.out.println("6. Copy the object to another Amazon S3 bucket.");
        copyBucketObject(s3, bucketName, key, toBucket);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the object from the Amazon S3 bucket.");
        deleteObjectFromBucket(s3, bucketName, key);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Delete the Amazon S3 bucket.");
        deleteBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully performed");
        System.out.println(DASHES);
        s3.close();
    }

    // Create a bucket by using a S3Waiter object.
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
public static void deleteBucket(S3Client client, String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    client.deleteBucket(deleteBucketRequest);
    System.out.println(bucket + " was deleted.");
}

/**
 * Upload an object in parts.
 */
public static void multipartUpload(S3Client s3, String bucketName, String key) {
    int mB = 1024 * 1024;
    // First create a multipart upload and get the upload id.
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

    CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
    String uploadId = response.uploadId();
    System.out.println(uploadId);

    // Upload all the different parts of the object.
    UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(1).build();

    String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
        .eTag();
    CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

    UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
        .uploadId(uploadId)
        .partNumber(2).build();
```



```
String etag2 = s3.uploadPart(uploadPartRequest2,
    RequestBody.fromByteBuffer(getRandomByteBuffer(3 * MB)))
    .eTag();
CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

// Call completeMultipartUpload operation to tell S3 to merge all uploaded
// parts and finish the multipart operation.
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
}

private static ByteBuffer getRandomByteBuffer(int size) {
    byte[] b = new byte[size];
    new Random().nextBytes(b);
    return ByteBuffer.wrap(b);
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
```

```
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
}

public static void listAllObjects(S3Client s3, String bucketName) {
    ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    boolean done = false;
    while (!done) {
        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
        for (S3Object content : listObjResponse.contents()) {
            System.out.println(content.key());
        }

        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()
            .continuationToken(listObjResponse.nextContinuationToken())
    }
```

```
        .build();
    }
}

public static void anotherListExample(S3Client s3, String bucketName) {
    ListObjectsV2Request listReq = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

    // Process response pages.
    listRes.stream()
        .flatMap(r -> r.contents().stream())
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

    // Helper method to work with paginated collection of items directly.
    listRes.contents().stream()
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

    for (S3Object content : listRes.contents()) {
        System.out.println(" Key: " + content.key() + " size = " +
content.size());
    }
}

public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.deleteObject(deleteObjectRequest);
    System.out.println(key + " was deleted");
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    String encodedUrl = null;
    try {
```

```
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .copySource(encodedUrl)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        System.out.println("The " + objectKey + " was copied to " + toBucket);
        return copyRes.copyObjectResult().toString();


    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Obtenir la configuration légale de conservation d'un objet

L'exemple de code suivant montre comment obtenir la configuration de conservation légale d'un compartiment S3.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "'");
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLegalHold](#) à la section Référence des AWS SDK for Java 2.x API.

Verrouiller des objets Amazon S3

L'exemple de code suivant montre comment utiliser les fonctionnalités de verrouillage d'objets S3.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les fonctionnalités de verrouillage d'objets d'Amazon S3.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
Before running this Java V2 code example, set up your development
environment, including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

This Java example performs the following tasks:
  1. Create test Amazon Simple Storage Service (S3) buckets with different lock
policies.
  2. Upload sample objects to each bucket.
  3. Set some Legal Hold and Retention Periods on objects and buckets.
  4. Investigate lock policies by viewing settings or attempting to delete or
overwrite objects.
  5. Clean up objects and buckets.
*/
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
```

```
private static final List<String> fileNames = new ArrayList<>();

public static void main(String[] args) {
    // Get the current date and time to ensure bucket name is unique.
    LocalDateTime currentTime = LocalDateTime.now();

    // Format the date and time as a string.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
    String timeStamp = currentTime.format(formatter);

    s3LockActions = new S3LockActions();
    bucketName = "bucket"+timeStamp;
    Scanner scanner = new Scanner(System.in);

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
Locking Workflow Scenario.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();
    configurationSetup();
    System.out.println(DASHES);

    System.out.println(DASHES);
    setup();
    System.out.println("Setup is complete. Press Enter to continue...");
    scanner.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Lets present the user with choices.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();
    demoActionChoices() ;
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Would you like to clean up the resources? (y/n)");
    String delAns = scanner.nextLine().trim();
    if (delAns.equalsIgnoreCase("y")) {
        cleanup();
        System.out.println("Clean up is complete.");
    }

    System.out.println("Press Enter to continue...");
```

```
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by selecting
one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                }
            }
        }
    }
}
```



```
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
    }

    case 2 -> {
        System.out.println("Enter the number of the object to delete:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        String version = allFiles.get(fileChoice).getVersion();
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
    }

    case 3 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();

        // Attempt to overwrite the file.
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
            writer.write("This is a modified text.");

        } catch (IOException e) {
            e.printStackTrace();
        }
        s3LockActions.uploadFile(bucketName, objectKey, objectKey);
    }

    case 4 -> {
```

```

        System.out.println("Enter the number of the object to
overwrite:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectRetention(bucketName, objectKey);
    }

    case 5 -> {
        System.out.println("Enter the number of the object to view:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectLegalHold(bucketName, objectKey);
        s3LockActions.getBucketObjectLockConfiguration(bucketName);
    }

    case 6 -> {
        System.out.println("Exiting the workflow...");
        return;
    }

    default -> {
        System.out.println("Invalid choice. Please select again.");
    }
}
}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {

```

```

        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key, false);
                }
            }
            // Check for a retention period.
            ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
            boolean hasRetentionPeriod ;
            hasRetentionPeriod = retention != null;
            s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

        } else {
            System.out.println(bucketName + " objects do not have a legal lock");
            s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
        }
    }

    // Delete the buckets.
    System.out.println("Delete "+bucketName);
    for (String bucket : bucketNames){
        s3LockActions.deleteBucketByName(bucket);
    }
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking features.
        """);
}

```

```
System.out.println("S3 buckets can be created either with or without object
lock enabled.");
System.out.println("Press Enter to continue...");
scanner.nextLine();

// Create three S3 buckets.
s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
System.out.println("Press Enter to continue.");
scanner.nextLine();

System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
object locking with a default retention period.");
s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
System.out.println("Press Enter to continue.");
scanner.nextLine();

System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
System.out.println("Press Enter to continue.");
scanner.nextLine();

// Upload some files to the buckets.
System.out.println("Now let's add some test files:");
String fileName = "exampleFile.txt";
int fileCount = 2;
try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
    writer.write("This is a sample file for uploading to a bucket.");

} catch (IOException e) {
    e.printStackTrace();
}

for (String bucketName : bucketNames){
    for (int i = 0; i < fileCount; i++) {
        // Get the file name without extension.
        String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
        int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
        if (extensionIndex > 0) {
```

```

        fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
    }

    // Create the numbered file names.
    String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
    fileNames.add(numberedFileName);
    s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
}
}

String question = null;
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println("Now we can set some object lock policies on individual
files:");
for (String bucketName : bucketNames) {
    for (int i = 0; i < fileNames.size(); i++){

        // No modifications to the objects in the first bucket.
        if (!bucketName.equals(bucketNames.get(0))) {
            String exampleFileName = fileNames.get(i);
            switch (i) {
                case 0 -> {
                    question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?"
;
                    System.out.println(question);
                    String ans = scanner.nextLine().trim();
                    if (ans.equalsIgnoreCase("y")) {
                        System.out.println("**** You have selected to put a
legal hold " + exampleFileName);

                            // Set a legal hold.
                            s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                    }
                }
                case 1 -> {
                    ""
                    Would you like to add a 1 day Governance retention
period to %s in %s (y/n)?

```

```

        Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
        ""formatted(exampleFileName, bucketName);
        System.out.println(question);
        String ans2 = scanner.nextLine().trim();
        if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
        }
    }
}

// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

public static void configurationSetup() {
    String noLockBucketName = bucketName + "-no-lock";
    String lockEnabledBucketName = bucketName + "-lock-enabled";
    String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
    bucketNames.add(noLockBucketName);
    bucketNames.add(lockEnabledBucketName);
    bucketNames.add(retentionAfterCreationBucketName);
}

public static int getChoiceResponse(String question, String[] choices) {
    Scanner scanner = new Scanner(System.in);
    if (question != null) {
        System.out.println(question);
        for (int i = 0; i < choices.length; i++) {
            System.out.println("\t" + (i + 1) + ". " + choices[i]);
        }
    }
}

```

```
int choiceNumber = 0;
while (choiceNumber < 1 || choiceNumber > choices.length) {
    String choice = scanner.nextLine();
    try {
        choiceNumber = Integer.parseInt(choice);
    } catch (NumberFormatException e) {
        System.out.println("Invalid choice. Please enter a valid number.");
    }
}

return choiceNumber - 1;
}
```

Une classe wrapper pour les fonctions S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
```

```
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time zone.
        ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);
    }
}
```



```
// Print the formatted date string.
System.out.println("Formatted Date: " + humanReadableDate);
ObjectLockRetention retention = ObjectLockRetention.builder()
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .retainUntilDate(futureInstant)
    .build();

PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

getClient().putObjectRetention(retentionRequest);
System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}

// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "'");
    }

    return null;
}
```

```
// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}

public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
    AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
    return bucketNames.stream()
        .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
        .map(version -> {
            S3InfoObject s3InfoObject = new S3InfoObject();
            s3InfoObject.setBucketName(bucketName);
            s3InfoObject.setVersion(version.versionId());
            s3InfoObject.setKeyName(version.key());
            return s3InfoObject;
        })))
        .peek(s3InfoObject -> {
            int i = counter.incrementAndGet(); // Increment and get the updated
value.

            if (interactive) {
                System.out.println(i + ": " + s3InfoObject.getKeyName());
                System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
            }
        })
        .collect(Collectors.toList());
}
```

```
}

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention retention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(retention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
```

```
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

    getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName +".");
}

// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
            .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on "+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage() +
""");
    }
}

public void uploadFile(String bucketName, String objectName, String filePath) {
    Path file = Paths.get(filePath);
    PutObjectRequest request = PutObjectRequest.builder()
```

```
        .bucket(bucketName)
        .key(objectName)
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
    }
}

// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
}

// Delete an object from a specific bucket.
public void deleteObjectFromBucket(String bucketName, String objectKey, boolean
hasRetention, String versionId) {
```

```
    try {
        DeleteObjectRequest objectRequest;
        if (hasRetention) {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .bypassGovernanceRetention(true)
                .build();
        } else {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .build();
        }

        getClient().deleteObject(objectRequest) ;
        System.out.println("The object was successfully deleted");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("Object retention for "+key+" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");

        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}
```

```
    }  
  }  
  
  public void deleteBucketByName(String bucketName) {  
    try {  
      DeleteBucketRequest request = DeleteBucketRequest.builder()  
        .bucket(bucketName)  
        .build();  
  
      getClient().deleteBucket(request);  
      System.out.println(bucketName + " was deleted.");  
  
    } catch (S3Exception e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
    }  
  }  
  
  // Get the object lock configuration details for an S3 bucket.  
  public void getBucketObjectLockConfiguration(String bucketName) {  
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =  
    GetObjectLockConfigurationRequest.builder()  
      .bucket(bucketName)  
      .build();  
  
    GetObjectLockConfigurationResponse response =  
    getClient().getObjectLockConfiguration(objectLockConfigurationRequest);  
    System.out.println("Bucket object lock config for "+bucketName +": ");  
    System.out.println("\tEnabled:  
"+response.getObjectLockConfiguration().getObjectLockEnabled());  
    System.out.println("\tRule: "+  
    response.getObjectLockConfiguration().rule().defaultRetention());  
  }  
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)

- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Analyse d'URI

L'exemple de code suivant montre comment analyser des URI Amazon S3 pour extraire des composants importants tels que le nom du compartiment et la clé d'objet.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Analysez un URI Amazon S3 à l'aide de la classe [S3Uri](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client - An S3Client through which you acquire an S3Uri instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate S3Uri
 * capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
    versionId=abc123&partNumber=77&partNumber=88'.
}
```



```
// Create an S3Utilities object using the configuration of the s3Client.
S3Utilities s3Utilities = s3Client.utilities();

// From a String URL create a URI object to pass to the parseUri() method.
URI uri = URI.create(s3ObjectUrl);
S3Uri s3Uri = s3Utilities.parseUri(uri);

// If the URI contains no value for the Region, bucket or key, the SDK
returns
// an empty Optional.
// The SDK returns decoded URI values.

Region region = s3Uri.region().orElse(null);
log("region", region);
// Console output: 'region: us-west-1'.

String bucket = s3Uri.bucket().orElse(null);
log("bucket", bucket);
// Console output: 'bucket: myBucket'.

String key = s3Uri.key().orElse(null);
log("key", key);
// Console output: 'key: resources/doc.txt'.

Boolean isPathStyle = s3Uri.isPathStyle();
log("isPathStyle", isPathStyle);
// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123], partNumber=[77,
// 88]}'

// Retrieve the first or all values for a query parameter as shown in the
// following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
```

```

// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
log("firstMatchingRawQueryParameter", partNumbers);
// Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

/*
 * Object keys and query parameters with reserved or unsafe characters, must
be
 * URL-encoded.
 * For example replace whitespace " " with "%20".
 * Valid:
 * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
 * Invalid:
 * "https://s3.us-west-1.amazonaws.com/myBucket/object key?query=[brackets]"
 *
 * Virtual-hosted-style URIs with bucket names that contain a dot, ".", the
dot
 * must not be URL-encoded.
 * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
 * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
 */
}

private static void log(String s3UriElement, Object element) {
    if (element == null) {
        logger.info("{}: {}", s3UriElement, "null");
    } else {
        logger.info("{}: {}", s3UriElement, element);
    }
}
}

```

Réalisation d'un chargement partitionné

L'exemple de code suivant montre comment effectuer un chargement partitionné vers un objet Amazon S3.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Les exemples de code utilisent les importations suivantes.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Utilisez le [gestionnaire de transferts S3](#) situé au-dessus du [client S3 basé sur AWS CRT](#) pour effectuer de manière transparente un téléchargement partitionné lorsque la taille du contenu dépasse un seuil. La taille par défaut est de 8 Mo.

```
public void multipartUploadWithTransferManager(String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Utilisez l'[API S3Client](#) pour effectuer un téléchargement en plusieurs parties.

```
public void multipartUploadWithS3Client(String filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

```

```
UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .partNumber(partNumber)
    .build();

UploadPartResponse partResponse = s3Client.uploadPart(
    uploadPartRequest,
    RequestBody.fromByteBuffer(bb));

CompletedPart part = CompletedPart.builder()
    .partNumber(partNumber)
    .eTag(partResponse.eTag())
    .build();
completedParts.add(part);

bb.clear();
position += read;
partNumber++;
}
} catch (IOException e) {
    logger.error(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}
```

Utilisez l'[AsyncClient API S3](#) avec le support multipartie activé pour effectuer un téléchargement partitionné.

```
public void multipartUploadWithS3AsyncClient(String filePath) {
    // Enable multipart support.
    S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
        .multipartEnabled(true)
```

```
        .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b ->
b            .bucket(bucketName)
                .key(key),
                Paths.get(filePath));


        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using S3AsyncClient.");
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Suivez les chargements et les téléchargements

L'exemple de code suivant montre comment suivre le chargement ou le téléchargement d'un objet Amazon S3.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Suivez la progression du téléchargement d'un fichier.

```
public void trackUploadFile(S3TransferManager transferManager, String
bucketName,

        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
```

```

        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .source(Paths.get(filePathURI))
        .build();

```

```
FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
```

```
fileUpload.completionFuture().join();
```

```
/*
```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure `log4j2` with settings such as the following.

```

<Configuration status="WARN">
  <Appenders>
    <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
  </Appenders>

  <Loggers>
    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
      <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
  </Loggers>
</Configuration>

```

`Log4j2` logs the progress. The following is example output for a 21.3 MB file upload.

```

Transfer initiated...
|                               | 0.0%
|====                          | 21.1%
|=====                        | 60.5%
|=====                        | 100.0%
Transfer complete!

```

```
*/
```

```
}

```

Suivez la progression du téléchargement d'un fichier.

```

public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                               String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .destination(Paths.get(downloadedFilePath))
        .build();

```

```

    FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

```

```

    CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
    /*

```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure `log4j2` with settings such as the following.

```

<Configuration status="WARN">
    <Appenders>
        <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%m%n"/>
        </Console>
    </Appenders>

    <Loggers>
        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
            <AppenderRef ref="AlignedConsoleAppender"/>
        </logger>
    </Loggers>
</Configuration>

```

`Log4j2` logs the progress. The following is example output for a 21.3 MB file download.

```

Transfer initiated...
|=====          | 39.4%
|=====          | 78.8%
|=====          | 100.0%

```



```
        Transfer complete!  
    */  
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [GetObject](#)
 - [PutObject](#)

Charger le répertoire dans un compartiment

L'exemple de code suivant montre comment charger un répertoire local de manière récursive dans un compartiment Amazon Simple Storage Service (Amazon S3).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez un [S3 TransferManager](#) pour [télécharger un répertoire local](#). Consultez le [fichier complet](#) et le [test](#).

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;  
import software.amazon.awssdk.transfer.s3.S3TransferManager;  
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;  
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;  
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;  
  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.net.URL;  
import java.nio.file.Paths;  
import java.util.UUID;
```

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}
```

- Pour plus de détails sur l'API, reportez-vous [UploadDirectory](#) à la section Référence des AWS SDK for Java 2.x API.

Charger ou télécharger des fichiers volumineux

L'exemple de code suivant montre comment charger ou télécharger des fichiers volumineux vers et depuis Amazon S3.

Pour plus d'informations, consultez la rubrique [Uploading an object using multipart upload](#) (Chargement d'un objet à l'aide du chargement partitionné).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Appelez des fonctions qui transfèrent des fichiers vers et depuis un compartiment S3 à l'aide du S3TransferManager.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
```

```

        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }

```

Chargez un répertoire local complet.

```

public Integer uploadDirectory(S3TransferManager transferManager,
        URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}

```

Chargez un seul fichier.

```

public String uploadFile(S3TransferManager transferManager, String bucketName,
        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();
}

```

```
FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
return uploadResult.response().eTag();
}
```

Charger un flux de taille inconnue

L'exemple de code suivant montre comment charger un flux de taille inconnue dans un objet Amazon S3.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez le [Client S3 basé sur CRT AWS](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * @param s3CrtAsyncClient - To upload content from a stream of unknown size,
 * use the AWS CRT-based S3 client. For more information, see
 * https://docs.aws.amazon.com/sdk-for-java/latest/
 * developer-guide/crt-based-s3-client.html.
 * @param bucketName - The name of the bucket.
```

```
    * @param key - The name of the object.
    * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
    metadata pertaining to the put object operation.
    */
    public PutObjectResponse putObjectFromStream(S3AsyncClient s3CrtAsyncClient,
    String bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
            AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
    stream will be provided later.

        CompletableFuture<PutObjectResponse> responseFuture =
            s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
    body);

        // AsyncExampleUtils.randomString() returns a random string up to 100
    characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
    randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        PutObjectResponse response = responseFuture.join(); // Wait for the
    response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        return response;
    }
}
```

Utilisez le [Gestionnaire de transferts Amazon S3](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;
```

```
import java.io.ByteArrayInputStream;
import java.util.UUID;

/**
 * @param transferManager - To upload content from a stream of unknown size, use
 the S3TransferManager based on the AWS CRT-based S3 client.
 *
 * For more information, see https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
 */
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    Upload upload = transferManager.upload(builder -> builder
        .requestBody(body)
        .putObjectRequest(req -> req.bucket(bucketName).key(key))
        .build());

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    return upload.completionFuture().join();
}
}
```

Utiliser les totaux de contrôle

L'exemple de code suivant montre comment utiliser des sommes de contrôle pour travailler avec un objet Amazon S3.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Les exemples de code utilisent un sous-ensemble des importations suivantes.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Spécifiez un algorithme de somme de contrôle pour la méthode `putObject` lorsque vous [créez l'élément `PutObjectRequest`](#).

```
public void putObjectWithChecksum() {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

Vérifiez la somme de contrôle de la `getObject` méthode lorsque vous [créez le `GetObjectRequest`](#).

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

Pré-calculez une somme de contrôle pour la méthode `putObject` lorsque vous [créez l'élément `PutObjectRequest`](#).

```
public void putObjectWithPrecalculatedChecksum(String filePath) {
    String checksum = calculateChecksum(filePath, "SHA-256");

    s3Client.putObject((b -> b
        .bucket(bucketName)
        .key(key)
        .checksumSHA256(checksum)),
```



```

        requestBody.fromFile(Paths.get(filePath));
    }

```

Utilisez le [gestionnaire de transferts S3](#) situé au-dessus du [client S3 basé sur AWS CRT](#) pour effectuer de manière transparente un téléchargement partitionné lorsque la taille du contenu dépasse un seuil. La taille par défaut est de 8 Mo.

Vous pouvez spécifier un algorithme de somme de contrôle que le kit SDK utilisera. Par défaut, le kit SDK utilise l'algorithme CRC32.

```

public void multipartUploadWithChecksumTm(String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key)
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}

```

Utilisez l'API [S3Client](#) ou ([AsyncClient API S3](#)) pour effectuer un téléchargement partitionné. Si vous spécifiez une somme de contrôle supplémentaire, vous devez spécifier l'algorithme à utiliser lors du lancement du chargement. Vous devez également spécifier l'algorithme pour chaque demande d'article et fournir la somme de contrôle calculée pour chaque article après son chargement.

```

public void multipartUploadWithChecksumS3Client(String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();
}

```

```
// Upload the parts of the file.
int partNumber = 1;
List<CompletedPart> completedParts = new ArrayList<>();
ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
    long fileSize = file.length();
    long position = 0;
    while (position < fileSize) {
        file.seek(position);
        long read = file.getChannel().read(bb);

        bb.flip(); // Swap position and limit before reading from the
buffer.

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .checksumAlgorithm(algorithm) // Checksum specified on each
part.

            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.

            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}
```

```
// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Exemples sans serveur

Invoquer une fonction lambda à partir d'un déclencheur Amazon S3

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par le téléchargement d'un objet dans un compartiment S3. La fonction extrait le nom du compartiment S3 et la clé de l'objet à partir du paramètre d'événement et appelle l'API Amazon S3 pour récupérer et consigner le type de contenu de l'objet.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement S3 avec Lambda en utilisant Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificationRecord;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

Exemples de S3 Glacier utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide de S3 Glacier.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateVault

L'exemple de code suivant montre comment utiliser `CreateVault`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateVault {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to create.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                .vaultName(vaultName)
                .build();

            CreateVaultResponse createVaultResult =
glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
createVaultResult.location());

        } catch (GlacierException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateVault](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteArchive

L'exemple de code suivant montre comment utiliser `DeleteArchive`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <vaultName> <accountId> <archiveId>
```

```
        Where:
            vaultName - The name of the vault that contains the archive to
delete.
            accountId - The account ID value.
            archiveId - The archive ID value.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String vaultName = args[0];
    String accountId = args[1];
    String archiveId = args[2];
    GlacierClient glacier = GlacierClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
    glacier.close();
}

public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
String accountId,
    String archiveId) {
    try {
        DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
            .vaultName(vaultName)
            .accountId(accountId)
            .archiveId(archiveId)
            .build();

        glacier.deleteArchive(delArcRequest);
        System.out.println("The archive was deleted.");

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```


- Pour plus de détails sur l'API, reportez-vous [DeleteArchive](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteVault

L'exemple de code suivant montre comment utiliser `DeleteVault`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteVault {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String vaultName = args[0];
    GlacierClient glacier = GlacierClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deleteGlacierVault(glacier, vaultName);
    glacier.close();
}

public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
    try {
        DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
            .vaultName(vaultName)
            .build();

        glacier.deleteVault(delVaultRequest);
        System.out.println("The vault was deleted!");

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteVault](#) à la section Référence des AWS SDK for Java 2.x API.

InitiateJob

L'exemple de code suivant montre comment utiliser `InitiateJob`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Récupérez l'inventaire d'un coffre.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName> <accountId> <path>

            Where:
                vaultName - The name of the vault.
                accountId - The account ID value.
                path - The path where the file is written to.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String vaultName = args[0];
String accountId = args[1];
String path = args[2];
GlacierClient glacier = GlacierClient.builder()
    .region(Region.US_EAST_1)
    .build();

String jobNum = createJob(glacier, vaultName, accountId);
checkJob(glacier, jobNum, vaultName, accountId, path);
glacier.close();
}

public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {
    try {
        JobParameters job = JobParameters.builder()
            .type("inventory-retrieval")
            .build();

        InitiateJobRequest initJob = InitiateJobRequest.builder()
            .jobParameters(job)
            .accountId(accountId)
            .vaultName(vaultName)
            .build();

        InitiateJobResponse response = glacier.initiateJob(initJob);
        System.out.println("The job ID is: " + response.jobId());
        System.out.println("The relative URI path of the job is: " +
response.location());
        return response.jobId();

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

// Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
// Documentation.
public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {
    try {
```

```
boolean finished = false;
String jobStatus;
int yy = 0;

while (!finished) {
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()
        .jobId(jobId)
        .accountId(account)
        .vaultName(name)
        .build();

    DescribeJobResponse response = glacier.describeJob(jobRequest);
    jobStatus = response.statusCodeAsString();

    if (jobStatus.compareTo("Succeeded") == 0)
        finished = true;
    else {
        System.out.println(yy + " status is: " + jobStatus);
        Thread.sleep(1000);
    }
    yy++;
}

System.out.println("Job has Succeeded");
GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
    .jobId(jobId)
    .vaultName(name)
    .accountId(account)
    .build();

ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
// Write the data to a local file.
byte[] data = objectBytes.asByteArray();
File myFile = new File(path);
OutputStream os = new FileOutputStream(myFile);
os.write(data);
System.out.println("Successfully obtained bytes from a Glacier vault");
os.close();

} catch (GlacierException | InterruptedException | IOException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
```

```
    }  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [InitiateJob](#) à la section Référence des AWS SDK for Java 2.x API.

ListVaults

L'exemple de code suivant montre comment utiliser `ListVaults`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;  
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;  
import software.amazon.awssdk.services.glacier.GlacierClient;  
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;  
import software.amazon.awssdk.services.glacier.model.GlacierException;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ListVaults {  
    public static void main(String[] args) {  
        GlacierClient glacier = GlacierClient.builder()  
            .region(Region.US_EAST_1)  
            .build();  
    }  
}
```

```
        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                        .marker(newMarker)
                        .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                        .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v : vaultList) {
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }

                // Check for further results.
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }
        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListVaults](#) à la section Référence des AWS SDK for Java 2.x API.

UploadArchive

L'exemple de code suivant montre comment utiliser `UploadArchive`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```



```
*/
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <strPath> <vaultName>\s

            Where:
                strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
                vaultName - The name of the vault.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                .vaultName(vaultName)
                .checksum(checkVal)
                .build();
```

```
        UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
        return res.archiveId();

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String computeSHA256(File inputFile) {
    try {
        byte[] treeHash = computeSHA256TreeHash(inputFile);
        System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
        return toHex(treeHash);

    } catch (IOException ioe) {
        System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
        System.exit(-1);

    } catch (NoSuchAlgorithmException nsae) {
        System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
        System.exit(-1);
    }
    return "";
}

public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
    NoSuchAlgorithmException {

    byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
    return computeSHA256TreeHash(chunkSHA256Hashes);
}

/**
 * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
 * includes the checksum for the last chunk, even if it's smaller than 1 MB.
 */
public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
    NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
```

```
    long numChunks = file.length() / ONE_MB;
    if (file.length() % ONE_MB > 0) {
        numChunks++;
    }

    if (numChunks == 0) {
        return new byte[][] { md.digest() };
    }

    byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
    FileInputStream fileStream = null;

    try {
        fileStream = new FileInputStream(file);
        byte[] buff = new byte[ONE_MB];

        int bytesRead;
        int idx = 0;

        while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
            md.reset();
            md.update(buff, 0, bytesRead);
            chunkSHA256Hashes[idx++] = md.digest();
        }

        return chunkSHA256Hashes;
    } finally {
        if (fileStream != null) {
            try {
                fileStream.close();
            } catch (IOException ioe) {
                System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                                ioe.getMessage());
            }
        }
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
```

```
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {
        int len = prevLvlHashes.length / 2;
        if (prevLvlHashes.length % 2 != 0) {
            len++;
        }

        byte[][] currLvlHashes = new byte[len][];
        int j = 0;
        for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

            // If there are at least two elements remaining.
            if (prevLvlHashes.length - i > 1) {

                // Calculate a digest of the concatenated nodes.
                md.reset();
                md.update(prevLvlHashes[i]);
                md.update(prevLvlHashes[i + 1]);
                currLvlHashes[j] = md.digest();

            } else { // Take care of the remaining odd chunk
                currLvlHashes[j] = prevLvlHashes[i];
            }
        }

        prevLvlHashes = currLvlHashes;
    }

    return prevLvlHashes[0];
}

/**
 * Returns the hexadecimal representation of the input byte array
 */
public static String toHex(byte[] data) {
    StringBuilder sb = new StringBuilder(data.length * 2);
    for (byte datum : data) {
        String hex = Integer.toHexString(datum & 0xFF);

        if (hex.length() == 1) {
```

```
        // Append leading zero.
        sb.append("0");
    }
    sb.append(hex);
}
return sb.toString().toLowerCase();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UploadArchive](#) à la section Référence des AWS SDK for Java 2.x API.

SageMaker exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with SageMaker.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour SageMaker

Les exemples de code suivants montrent comment commencer à utiliser SageMaker.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
            .region(region)
            .build();

        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
item.notebookInstanceName());
            }

        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListNotebookInstances](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreatePipeline

L'exemple de code suivant montre comment utiliser `CreatePipeline`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
        System.out.println(jsonObject);

        // Create the pipeline.
        CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
            .pipelineDescription("Java SDK example pipeline")
            .roleArn(roleArn)
```

```
        .pipelineName(pipelineName)
        .pipelineDefinition(jsonObject.toString())
        .build();

    sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreatePipeline](#) à la section Référence des AWS SDK for Java 2.x API.

DeletePipeline

L'exemple de code suivant montre comment utiliser `DeletePipeline`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("*** Successfully deleted " + pipelineName);
}
```


- Pour plus de détails sur l'API, reportez-vous [DeletePipeline](#) à la section Référence des AWS SDK for Java 2.x API.

DescribePipelineExecution

L'exemple de code suivant montre comment utiliser `DescribePipelineExecution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
    int index = 0;
    do {
        DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
            .pipelineExecutionArn(executionArn)
            .build();

        DescribePipelineExecutionResponse response = sageMakerClient
            .describePipelineExecution(pipelineExecutionRequest);
        status = response.pipelineExecutionStatusAsString();
        System.out.println(index + ". The Status of the pipeline is " + status);
        TimeUnit.SECONDS.sleep(4);
        index++;
    } while ("Executing".equals(status));
    System.out.println("Pipeline finished with status " + status);
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribePipelineExecution](#) à la section Référence des AWS SDK for Java 2.x API.

StartPipelineExecution

L'exemple de code suivant montre comment utiliser `StartPipelineExecution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
```

```
        " },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";

System.out.println(inputJSON);

Parameter para3 = Parameter.builder()
    .name("parameter_vej_input_config")
    .value(inputJSON)
    .build();

// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();
System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
```

```
        .value(para5JSON)
        .build();

System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}
```

- Pour plus de détails sur l'API, reportez-vous [StartPipelineExecution](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Commencez par les travaux et les pipelines géospatiaux

L'exemple de code suivant illustre comment :

- Configurez les ressources d'un pipeline.
- Configurez un pipeline qui exécute une tâche géospatiale.
- Démarrez l'exécution d'un pipeline.
- Surveillez le statut de l'exécution.
- Affichez la sortie du pipeline.
- Nettoyez les ressources.

Pour plus d'informations, voir [Création et exécution de SageMaker pipelines à l'aide de kits de AWS développement logiciel \(SDK\) sur Community.aws](#).

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
<functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
<pipelineName>\n\n"
            +
            "Where:\n" +
            "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
            +
            "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
            "    functionFileLocation - The file location where the JAR file
that represents the AWS Lambda function is located.\n\n"
            +
            "    functionName - The name of the AWS Lambda function (for
example, SageMakerExampleFunction).\n\n" +
            "    queueName - The name of the Amazon Simple Queue Service (Amazon
SQS) queue.\n\n" +
            "    bucketName - The name of the Amazon Simple Storage Service
(Amazon S3) bucket.\n\n" +
            "    lnglatData - The file location of the latlongtest.csv file
required for this use case.\n\n" +
            "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"
            +
            "    pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).\n\n";
```

```
if (args.length != 9) {
    System.out.println(usage);
    System.exit(1);
}

String sagemakerRoleName = args[0];
String lambdaRoleName = args[1];
String functionFileLocation = args[2];
String functionName = args[3];
String queueName = args[4];
String bucketName = args[5];
String lnglatData = args[6];
String spatialPipelinePath = args[7];
String pipelineName = args[8];
String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

Region region = Region.US_WEST_2;
SageMakerClient sagemakerClient = SageMakerClient.builder()
    .region(region)
    .build();

IamClient iam = IamClient.builder()
    .region(region)
    .build();

LambdaClient lambdaClient = LambdaClient.builder()
    .region(region)
    .build();

SqsClient sqsClient = SqsClient.builder()
    .region(region)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
System.out.println(
    "\nThis example workflow will guide you through setting up and
running an" +
```

```
        "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
        "\nreverse geocode addresses in an input file and store the
results in an export file.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
    String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
    String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

    String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
        handlerName);
    String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
    System.out.println("The queue URL is " + queueUrl);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Setting up bucket " + bucketName);
    if (!checkBucket(s3Client, bucketName)) {
        setupBucket(s3Client, bucketName);
        System.out.println("Put " + lnglatData + " into " + bucketName);
        putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Now we can create and run our pipeline.");
    setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
    String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
        pipelineName);
    System.out.println("The pipeline execution ARN value is " +
pipelineExecutionARN);
    waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
    System.out.println("Getting output results " + bucketName);
    getOutputResults(s3Client, bucketName);
    System.out.println(DASHES);
```

```

        System.out.println(DASHES);
        System.out.println("The pipeline has completed. To view the pipeline and
runs " +
            "in SageMaker Studio, follow these instructions:" +
            "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }

    private static void readObject(S3Client s3Client, String bucketName, String key)
    {
        System.out.println("Output file contents: \n");
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
    }

```



```
        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] byteArray = objectBytes.asByteArray();
        String text = new String(byteArray, StandardCharsets.UTF_8);
        System.out.println("Text output: " + text);
    }

    // Display some results from the output directory.
    public static void getOutputResults(S3Client s3Client, String bucketName) {
        System.out.println("Getting output results {" + bucketName + "}");
        ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
            .bucket(bucketName)
            .prefix("outputfiles/")
            .build();

        ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
        List<S3Object> s3Objects = response.contents();
        for (S3Object object : s3Objects) {
            readObject(s3Client, bucketName, object.key());
        }
    }

    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
        throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                .pipelineExecutionArn(executionArn)
                .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " + status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }
}
```

```
// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("*** Successfully deleted " + pipelineName);
}

// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
        System.out.println(jsonObject);

        // Create the pipeline.
        CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
            .pipelineDescription("Java SDK example pipeline")
            .roleArn(roleArn)
            .pipelineName(pipelineName)
            .pipelineDefinition(jsonObject.toString())
            .build();

        sageMakerClient.createPipeline(pipelineRequest);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```

    } catch (IOException | ParseException e) {
        throw new RuntimeException(e);
    }
}

// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://\" + bucketName + \"/samplefiles/
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
        "  },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";

    System.out.println(inputJSON);

    Parameter para3 = Parameter.builder()
        .name("parameter_vej_input_config")

```

```
        .value(inputJSON)
        .build();

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
        .s3Uri(output)
        .build();

    ExportVectorEnrichmentJobOutputConfig outputConfig =
    ExportVectorEnrichmentJobOutputConfig.builder()
        .s3Data(jobS3Data)
        .build();

    String gson4 = gson.toJson(outputConfig);
    Parameter para4 = Parameter.builder()
        .name("parameter_vej_export_config")
        .value(gson4)
        .build();
    System.out.println("parameter_vej_export_config:" +
    gson.toJson(outputConfig));

    // Create a VectorEnrichmentJobConfig object.
    ReverseGeocodingConfig reverseGeocodingConfig =
    ReverseGeocodingConfig.builder()
        .xAttributeName("Longitude")
        .yAttributeName("Latitude")
        .build();

    VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
        .reverseGeocodingConfig(reverseGeocodingConfig)
        .build();

    String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
    {\"XAttributeName\": \"Longitude\", \"YAttributeName\": \"Latitude\"}}";
    Parameter para5 = Parameter.builder()
        .name("parameter_step_1_vej_config")
        .value(para5JSON)
        .build();

    System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
    parameters.add(para1);
    parameters.add(para2);
    parameters.add(para3);
    parameters.add(para4);
```

```
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}

public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
    DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
    .uuid(eventSourceMapping)
    .build();

    lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
}

public static void deleteSagemakerRole(IamClient iam, String roleName) {
    String[] sageMakerRolePolicies = getSageMakerRolePolicies();
    try {
        for (String policy : sageMakerRolePolicies) {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy)
                .roleName(roleName)
                .build();

            iam.detachRolePolicy(rolePolicyRequest);
        }

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
    }
}
```

```
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    try {
        for (String policy : lambdaRolePolicies) {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy)
                .roleName(roleName)
                .build();

            iam.detachRolePolicy(rolePolicyRequest);
        }

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the specific AWS Lambda function.
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();
```

```
        awsLambda.deleteFunction(request);
        System.out.println("*** " + functionName + " was deleted");

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Delete the specific S3 bucket.
public static void deleteBucket(S3Client s3Client, String bucketName) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
    s3Client.deleteBucket(deleteBucketRequest);
    System.out.println("*** " + bucketName + " was deleted.");
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            deleteBucketObjects(s3, bucketName, myValue.key());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
    toDelete.add(ObjectIdentifier.builder()
        .key(objectName)
        .build());
}
```

```
    try {
        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        s3.deleteObjects(dor);
        System.out.println("*** " + bucketName + " objects were deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the specific Amazon SQS queue.
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key("samplefiles/" + objectKey)
```



```
        .metadata(metadata)
        .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void setupBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Set up the SQS queue to use with the pipeline.
public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Setting up queue named " + queueName);
    try {
        Map<QueueAttributeName, String> queueAtt = new HashMap<>();
```

```

        queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
        queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
        queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .attributes(queueAtt)
            .build();

        sqsClient.createQueue(createQueueRequest);
        System.out.println("\nGet queue url");
        GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        TimeUnit.SECONDS.sleep(15);

        connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
        System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

// Connect the queue to the Lambda function as an event source.
public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
    String lambdaName) {
    System.out.println("Connecting the Lambda function and queue for the
pipeline.");
    String queueArn = "";

    // Specify the attributes to retrieve.
    List<QueueAttributeName> atts = new ArrayList<>();
    atts.add(QueueAttributeName.QUEUE_ARN);
    GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)

```

```
        .attributeNames(atts)
        .build();

    GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
    Map<String, String> queueAtts = response.attributesAsStrings();
    for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
        System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
        queueArn = queueAtt.getValue();
    }

    CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
        .eventSourceArn(queueArn)
        .functionName(lambdaName)
        .build();

    CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
    eventSourceMapping = response1.uuid();
    System.out.println("The mapping between the event source and Lambda function
was successful");
}

// Create an AWS Lambda function.
public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
String handler) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("SageMaker example function.")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .timeout(200)
```

```

        .memorySize(1024)
        .role(role)
        .build();

    // Create a Lambda function using a waiter.
    CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
    GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
        .functionName(functionName)
        .build();
    WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("The function ARN is " +
functionResponse.functionArn());
    return functionResponse.functionArn();

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String createSageMakerRole(IamClient iam, String roleName) {
    String[] sageMakerRolePolicies = getSageMakerRolePolicies();
    System.out.println("Creating a role to use with SageMaker.");
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\", " +
        "\"sagemaker-geospatial.amazonaws.com\", " +
        "\"lambda.amazonaws.com\", " +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}, " +
        "\"Action\": \"sts:AssumeRole\"" +
        "]]" +
        "}";

    try {

```

```
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)
            .assumeRolePolicyDocument(assumeRolePolicy)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse roleResult = iam.createRole(request);

        // Attach the policies to the role.
        for (String policy : sageMakerRolePolicies) {
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policy)
                .build();

            iam.attachRolePolicy(attachRequest);
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15);
        System.out.println("Role ready with ARN " + roleResult.role().arn());
        return roleResult.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

private static String createLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
```

```
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "]" +
        "};

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)
            .assumeRolePolicyDocument(assumeRolePolicy)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse roleResult = iam.createRole(request);

        // Attach the policies to the role.
        for (String policy : lambdaRolePolicies) {
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policy)
                .build();

            iam.attachRolePolicy(attachRequest);
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15);
        System.out.println("Role ready with ARN " + roleResult.role().arn());
        return roleResult.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());

    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
String handler) {
```

```
System.out.println("Create an AWS Lambda function used in this workflow.");
String functionArn;
try {
    // Does this function already exist.
    GetFunctionRequest functionRequest = GetFunctionRequest.builder()
        .functionName(functionName)
        .build();

    GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
    functionArn = response.configuration().functionArn();

} catch (LambdaException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
}
return functionArn;
}

// Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
// method returns true.
public static boolean checkBucket(S3Client s3, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3.headBucket(headBucketRequest);
        System.out.println(bucketName + " exists");
        return true;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a
// new queue
// and returns the ARN value.
public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
String lambdaName) {
```

```
System.out.println("Creating a queue for this use case.");
String queueUrl;
try {
    GetQueueUrlRequest request = GetQueueUrlRequest.builder()
        .queueName(queueName)
        .build();

    GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
    queueUrl = response.queueUrl();
    System.out.println(queueUrl);

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
}
return queueUrl;
}

// Checks to see if the Lambda role exists. If not, this method creates it.
public static String checkLambdaRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS Lambda to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createLambdaRole(iam, roleName);
    }
    return roleArn;
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
public static String checkSageMakerRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS SageMaker to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
```



```

        .roleName(roleName)
        .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createSageMakerRole(iam, roleName);
    }
    return roleArn;
}

private static String[] getSageMakerRolePolicies() {
    String[] sageMakerRolePolicies = new String[3];
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
    "AmazonSageMakerGeospatialFullAccess";
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    return sageMakerRolePolicies;
}

private static String[] getLambdaRolePolicies() {
    String[] lambdaRolePolicies = new String[5];
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
    "AmazonSageMakerGeospatialFullAccess";
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
        + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
    "AWSLambdaSQSQueueExecutionRole";
    return lambdaRolePolicies;
}
}

```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreatePipeline](#)

- [DeletePipeline](#)
- [DescribePipelineExecution](#)
- [StartPipelineExecution](#)
- [UpdatePipeline](#)

Exemples de Secrets Manager utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with Secrets Manager.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

GetSecretValue

L'exemple de code suivant montre comment utiliser `GetSecretValue`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <secretName>\s

                Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
```

```
        .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }

    public static void getValue(SecretsManagerClient secretsClient, String
secretName) {
        try {
            GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

            GetSecretValueResponse valueResponse =
secretsClient.getSecretValue(valueRequest);
            String secret = valueResponse.secretString();
            System.out.println(secret);

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetSecretValue](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples Amazon SES utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon SES.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

ListIdentities

L'exemple de code suivant montre comment utiliser `ListIdentities`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
            .region(region)
```

```
        .build();

    listSESIIdentities(client);
}

public static void listSESIIdentities(SesClient client) {
    try {
        ListIdentitiesResponse identitiesResponse = client.listIdentities();
        List<String> identities = identitiesResponse.identities();
        for (String identity : identities) {
            System.out.println("The identity is " + identity);
        }
    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListIdentities](#) à la section Référence des AWS SDK for Java 2.x API.

ListTemplates

L'exemple de code suivant montre comment utiliser `ListTemplates`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
```

```
public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        listAllTemplates(sesv2Client);
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
                ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();

            ListEmailTemplatesResponse response =
                sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                .forEach(template -> System.out.println("Template name: " +
                    template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTemplates](#) à la section Référence des AWS SDK for Java 2.x API.

SendEmail

L'exemple de code suivant montre comment utiliser `SendEmail`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s

            """;

        if (args.length != 3) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String sender = args[0];
    String recipient = args[1];
    String subject = args[2];

    Region region = Region.US_EAST_1;
    SesClient client = SesClient.builder()
        .region(region)
        .build();

    // The HTML body of the email.
    String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
        + "<p> See the list of customers.</p>" + "</body>" + "</html>";

    try {
        send(client, sender, recipient, subject, bodyHTML);
        client.close();
        System.out.println("Done");
    } catch (MessagingException e) {
        e.printStackTrace();
    }
}

public static void send(SesClient client,
    String sender,
    String recipient,
    String subject,
    String bodyHTML) throws MessagingException {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();
```

```
        Body body = Body.builder()
            .html(content)
            .build();

        Message msg = Message.builder()
            .subject(sub)
            .body(body)
            .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
            .destination(destination)
            .message(msg)
            .source(sender)
            .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
```

```
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject> <fileLocation>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
                fileLocation - The location of a Microsoft Excel file to use as
an attachment (C:/AWS/customers.xls).\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
            + "of customers to contact.";
```

```
// The HTML body of the email.
String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
    + "<p>Please see the attached file for a " + "list of customers to
contact.</p>" + "</body>"
    + "</html>";

Region region = Region.US_WEST_2;
SesClient client = SesClient.builder()
    .region(region)
    .build();

try {
    sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
    client.close();
    System.out.println("Done");
} catch (IOException | MessagingException e) {
    e.printStackTrace();
}
}

public static void sendemailAttachment(SesClient client,
    String sender,
    String recipient,
    String subject,
    String bodyText,
    String bodyHTML,
    String fileLocation) throws AddressException, MessagingException,
IOException {

    java.io.File theFile = new java.io.File(fileLocation);
    byte[] fileContent = Files.readAllBytes(theFile.toPath());

    Session session = Session.getDefaultInstance(new Properties());

    // Create a new MimeMessage object.
    MimeMessage message = new MimeMessage(session);

    // Add subject, from and to lines.
    message.setSubject(subject, "UTF-8");
    message.setFrom(new InternetAddress(sender));
    message.setRecipients(Message.RecipientType.TO,
    InternetAddress.parse(recipient));
```

```
// Create a multipart/alternative child container.
MimeMultipart msgBody = new MimeMultipart("alternative");

// Create a wrapper for the HTML and text parts.
MimeBodyPart wrap = new MimeBodyPart();

// Define the text part.
MimeBodyPart textPart = new MimeBodyPart();
textPart.setContent(bodyText, "text/plain; charset=UTF-8");

// Define the HTML part.
MimeBodyPart htmlPart = new MimeBodyPart();
htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

// Add the text and HTML parts to the child container.
msgBody.addBodyPart(textPart);
msgBody.addBodyPart(htmlPart);

// Add the child container to the wrapper object.
wrap.setContent(msgBody);

// Create a multipart/mixed parent container.
MimeMultipart msg = new MimeMultipart("mixed");

// Add the parent container to the message.
message.setContent(msg);
msg.addBodyPart(wrap);

// Define the attachment.
MimeBodyPart att = new MimeBodyPart();
DataSource fds = new ByteArrayDataSource(fileContent,
    "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
att.setDataHandler(new DataHandler(fds));

String reportName = "WorkReport.xls";
att.setFileName(reportName);

// Add the attachment to the message.
msg.addBodyPart(att);

try {
```

```
        System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");

        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        message.writeTo(outputStream);

        ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

        byte[] arr = new byte[buf.remaining()];
        buf.get(arr);

        SdkBytes data = SdkBytes.fromByteArray(arr);
        RawMessage rawMessage = RawMessage.builder()
            .data(data)
            .build();

        SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
            .rawMessage(rawMessage)
            .build();

        client.sendRawEmail(rawEmailRequest);

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Email sent using SesClient with attachment");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [SendEmail](#) à la section Référence des AWS SDK for Java 2.x API.

SendTemplatedEmail

L'exemple de code suivant montre comment utiliser `SendTemplatedEmail`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <template> <sender> <recipient>\s

            Where:
                template - The name of the email template.
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s

        """;
```

```
    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String templateName = args[0];
    String sender = args[1];
    String recipient = args[2];
    Region region = Region.US_EAST_1;
    SesV2Client sesv2Client = SesV2Client.builder()
        .region(region)
        .build();

    send(sesv2Client, sender, recipient, templateName);
}

public static void send(SesV2Client client, String sender, String recipient,
String templateName) {
    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    /*
     * Specify both name and favorite animal (favoriteanimal) in your code when
     * defining the Template object.
     * If you don't specify all the variables in the template, Amazon SES
doesn't
     * send the email.
     */
    Template myTemplate = Template.builder()
        .templateName(templateName)
        .templateData("{\n" +
            "  \"name\": \"Jason\"\n," +
            "  \"favoriteanimal\": \"Cat\"\n" +
            "}")
        .build();

    EmailContent emailContent = EmailContent.builder()
        .template(myTemplate)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
```



```
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email based on a template
using the AWS SDK for Java (v2)...");
        client.sendEmail(emailRequest);
        System.out.println("email based on a template was sent");

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SendTemplatedEmail](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'API Amazon SES v2 utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de l' AWS SDK for Java 2.x API v2 d'Amazon SES.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateContact

L'exemple de code suivant montre comment utiliser `CreateContact`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    // Create a new contact with the provided email address in the
    CreateContactRequest contactRequest = CreateContactRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .emailAddress(emailAddress)
        .build();

    sesClient.createContact(contactRequest);
    contacts.add(emailAddress);

    System.out.println("Contact created: " + emailAddress);

    // Send a welcome email to the new contact
    String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
    String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

    SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
        .fromEmailAddress(this.verifiedEmail)
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .simple(
                Message.builder()
                    .subject(Content.builder().data("Welcome to the Weekly
Coupons Newsletter").build())
                    .body(Body.builder()
                        .text(Content.builder().data(welcomeText).build())
                        .html(Content.builder().data(welcomeHtml).build())
```

```
        .build())
        .build())
        .build())
        .build();
    SendEmailResponse welcomeEmailResponse =
sesClient.sendEmail(welcomeEmailRequest);
    System.out.println("Welcome email sent: " +
welcomeEmailResponse.messageId());
    } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
        proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
    } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
emailAddress + ": " + e.getMessage());
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateContact](#) à la section Référence des AWS SDK for Java 2.x API.

CreateContactList

L'exemple de code suivant montre comment utiliser `CreateContactList`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    // 2. Create a contact list
    String contactListName = CONTACT_LIST_NAME;
    CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
```

```
        .contactListName(contactListName)
        .build();
    sesClient.createContactList(createContactListRequest);
    System.out.println("Contact list created: " + contactListName);
} catch (AlreadyExistsException e) {
    System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
} catch (LimitExceededException e) {
    System.err.println("Limit for contact lists has been exceeded.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating contact list: " + e.getMessage());
    throw e;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateContactList](#) à la section Référence des AWS SDK for Java 2.x API.

CreateEmailIdentity

L'exemple de code suivant montre comment utiliser `CreateEmailIdentity`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
    .emailIdentity(verifiedEmail)
    .build();
    sesClient.createEmailIdentity(createEmailIdentityRequest);
    System.out.println("Email identity created: " + verifiedEmail);
} catch (AlreadyExistsException e) {
    System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
}
```

```
    } catch (NotFoundException e) {
        System.err.println("The provided email address is not verified: " +
verifiedEmail);
        throw e;
    } catch (LimitExceededException e) {
        System.err
            .println("You have reached the limit for email identities. Please remove
some identities and try again.");
        throw e;
    } catch (SesV2Exception e) {
        System.err.println("Error creating email identity: " + e.getMessage());
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateEmailIdentity](#) à la section Référence des AWS SDK for Java 2.x API.

CreateEmailTemplate

L'exemple de code suivant montre comment utiliser `CreateEmailTemplate`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    // Create an email template named "weekly-coupons"
    String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
    String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

    CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
        .templateName(TEMPLATE_NAME)
        .templateContent(EmailTemplateContent.builder()
```

```
        .subject("Weekly Coupons Newsletter")
        .html(newsletterHtml)
        .text(newsletterText)
        .build()
    .build();

    sesClient.createEmailTemplate(templateRequest);

    System.out.println("Email template created: " + TEMPLATE_NAME);
} catch (AlreadyExistsException e) {
    // If the template already exists, skip this step and proceed with the next
    // operation
    System.out.println("Email template already exists, skipping creation...");
} catch (LimitExceededException e) {
    // If the limit for email templates is exceeded, fail the workflow and inform
    // the user
    System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
    throw e;
} catch (Exception e) {
    System.err.println("Error occurred while creating email template: " +
e.getMessage());
    throw e;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateEmailTemplate](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteContactList

L'exemple de code suivant montre comment utiliser `DeleteContactList`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    // Delete the contact list
    DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

    sesClient.deleteContactList(deleteContactListRequest);

    System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
} catch (NotFoundException e) {
    // If the contact list does not exist, log the error and proceed
    System.out.println("Contact list not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
    e.printStackTrace();
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteContactList](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteEmailIdentity

L'exemple de code suivant montre comment utiliser `DeleteEmailIdentity`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    // Delete the email identity
    DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
    .emailIdentity(this.verifiedEmail)
    .build();
```

```
sesClient.deleteEmailIdentity(deleteIdentityRequest);

System.out.println("Email identity deleted: " + this.verifiedEmail);
} catch (NotFoundException e) {
    // If the email identity does not exist, log the error and proceed
    System.out.println("Email identity not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
    e.printStackTrace();
}
} else {
    System.out.println("Skipping email identity deletion.");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteEmailIdentity](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteEmailTemplate

L'exemple de code suivant montre comment utiliser `DeleteEmailTemplate`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    // Delete the template
    DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
    .templateName(TEMPLATE_NAME)
    .build();

    sesClient.deleteEmailTemplate(deleteTemplateRequest);
}
```



```
        System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
        // If the email template does not exist, log the error and proceed
        System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
        System.err.println("Error occurred while deleting the email template: " +
            e.getMessage());
        e.printStackTrace();
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteEmailTemplate](#) à la section Référence des AWS SDK for Java 2.x API.

ListContacts

L'exemple de code suivant montre comment utiliser `ListContacts`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
ListContactsRequest contactListRequest = ListContactsRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

List<String> contactEmails;
try {
    ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);

    contactEmails = contactListResponse.contacts().stream()
        .map(Contact::emailAddress)
        .toList();
} catch (Exception e) {
    // TODO: Remove when listContacts's GET body issue is resolved.
    contactEmails = this.contacts;
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [ListContacts](#) à la section Référence des AWS SDK for Java 2.x API.

SendEmail

L'exemple de code suivant montre comment utiliser `SendEmail`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Envoie un message.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <sender> <recipient> <subject>\s

    Where:
        sender - An email address that represents the
sender.\s
        recipient - An email address that represents the
recipient.\s
        subject - The subject line.\s
"";

if (args.length != 3) {
    System.out.println(usage);
    System.exit(1);
}

String sender = args[0];
String recipient = args[1];
String subject = args[2];

Region region = Region.US_EAST_1;
SesV2Client sesv2Client = SesV2Client.builder()
    .region(region)
    .build();

// The HTML body of the email.
String bodyHTML = "<html>" + "<head></head>" + "<body>" +
"<h1>Hello!</h1>"
    + "<p> See the list of customers.</p>" + "</body>" +
"</html>";

    send(sesv2Client, sender, recipient, subject, bodyHTML);
}

public static void send(SesV2Client client,
    String sender,
    String recipient,
    String subject,
    String bodyHTML) {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
```

```
        .build();

        Content content = Content.builder()
            .data(bodyHTML)
            .build();

        Content sub = Content.builder()
            .data(subject)
            .build();

        Body body = Body.builder()
            .html(content)
            .build();

        Message msg = Message.builder()
            .subject(sub)
            .body(body)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(msg)
            .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
            .destination(destination)
            .content(emailContent)
            .fromEmailAddress(sender)
            .build();

        try {
            System.out.println("Attempting to send an email through
Amazon SES "
                               + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Envoie un message à l'aide d'un modèle.

```
String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
for (String emailAddress : contactEmails) {
    SendEmailRequest newsletterRequest = SendEmailRequest.builder()
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .template(Template.builder()
                .templateName(TEMPLATE_NAME)
                .templateData(coupons)
                .build())
            .build())
        .fromEmailAddress(this.verifiedEmail)
        .listManagementOptions(ListManagementOptions.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build())
        .build();
    SendEmailResponse newsletterResponse =
sesClient.sendEmail(newsletterRequest);
    System.out.println("Newsletter sent to " + emailAddress + ": " +
newsletterResponse.messageId());
}
```

- Pour plus de détails sur l'API, reportez-vous [SendEmail](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Flux de travail des newsletters

L'exemple de code suivant montre comment exécuter le flux de newsletter de l'API Amazon SES v2.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    // 2. Create a contact list
    String contactListName = CONTACT_LIST_NAME;
    CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
    .contactListName(contactListName)
    .build();
    sesClient.createContactList(createContactListRequest);
    System.out.println("Contact list created: " + contactListName);
} catch (AlreadyExistsException e) {
    System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
} catch (LimitExceededException e) {
    System.err.println("Limit for contact lists has been exceeded.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating contact list: " + e.getMessage());
    throw e;
}

try {
    // Create a new contact with the provided email address in the
    CreateContactRequest contactRequest = CreateContactRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .emailAddress(emailAddress)
    .build();

    sesClient.createContact(contactRequest);
    contacts.add(emailAddress);

    System.out.println("Contact created: " + emailAddress);

    // Send a welcome email to the new contact
    String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
    String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

    SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
    .fromEmailAddress(this.verifiedEmail)
    .destination(Destination.builder().toAddresses(emailAddress).build())
    .content(EmailContent.builder()
    .simple(
```

```
        Message.builder()
            .subject(Content.builder().data("Welcome to the Weekly
Coupons Newsletter").build())
            .body(Body.builder()
                .text(Content.builder().data(welcomeText).build())
                .html(Content.builder().data(welcomeHtml).build())
                .build())
            .build()
        .build();
        SendEmailResponse welcomeEmailResponse =
sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
welcomeEmailResponse.messageId());
    } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
    } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
emailAddress + ": " + e.getMessage());
        throw e;
    }
}

ListContactsRequest contactListRequest = ListContactsRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

List<String> contactEmails;
try {
    ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);

    contactEmails = contactListResponse.contacts().stream()
        .map(Contact::emailAddress)
        .toList();
} catch (Exception e) {
    // TODO: Remove when listContacts's GET body issue is resolved.
    contactEmails = this.contacts;
}
```

```
String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
for (String emailAddress : contactEmails) {
    SendEmailRequest newsletterRequest = SendEmailRequest.builder()
        .destination(Destination.builder().toAddresses(emailAddress).build())
        .content(EmailContent.builder()
            .template(Template.builder()
                .templateName(TEMPLATE_NAME)
                .templateData(coupons)
                .build())
            .build())
        .fromEmailAddress(this.verifiedEmail)
        .listManagementOptions(ListManagementOptions.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build())
        .build();
    SendEmailResponse newsletterResponse =
sesClient.sendEmail(newsletterRequest);
    System.out.println("Newsletter sent to " + emailAddress + ": " +
newsletterResponse.messageId());
}

try {
    CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
        .emailIdentity(verifiedEmail)
        .build();
    sesClient.createEmailIdentity(createEmailIdentityRequest);
    System.out.println("Email identity created: " + verifiedEmail);
} catch (AlreadyExistsException e) {
    System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
} catch (NotFoundException e) {
    System.err.println("The provided email address is not verified: " +
verifiedEmail);
    throw e;
} catch (LimitExceededException e) {
    System.err
        .println("You have reached the limit for email identities. Please remove
some identities and try again.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating email identity: " + e.getMessage());
    throw e;
}
```



```
}

try {
    // Create an email template named "weekly-coupons"
    String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
    String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

    CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
        .templateName(TEMPLATE_NAME)
        .templateContent(EmailTemplateContent.builder()
            .subject("Weekly Coupons Newsletter")
            .html(newsletterHtml)
            .text(newsletterText)
            .build())
        .build();

    sesClient.createEmailTemplate(templateRequest);

    System.out.println("Email template created: " + TEMPLATE_NAME);
} catch (AlreadyExistsException e) {
    // If the template already exists, skip this step and proceed with the next
    // operation
    System.out.println("Email template already exists, skipping creation...");
} catch (LimitExceededException e) {
    // If the limit for email templates is exceeded, fail the workflow and inform
    // the user
    System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
    throw e;
} catch (Exception e) {
    System.err.println("Error occurred while creating email template: " +
e.getMessage());
    throw e;
}

try {
    // Delete the contact list
    DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .build();
```

```
sesClient.deleteContactList(deleteContactListRequest);

System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
} catch (NotFoundException e) {
    // If the contact list does not exist, log the error and proceed
    System.out.println("Contact list not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
    e.printStackTrace();
}

try {
    // Delete the email identity
    DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
        .emailIdentity(this.verifiedEmail)
        .build();

    sesClient.deleteEmailIdentity(deleteIdentityRequest);

    System.out.println("Email identity deleted: " + this.verifiedEmail);
} catch (NotFoundException e) {
    // If the email identity does not exist, log the error and proceed
    System.out.println("Email identity not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
    e.printStackTrace();
}
} else {
    System.out.println("Skipping email identity deletion.");
}

try {
    // Delete the template
    DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
        .templateName(TEMPLATE_NAME)
        .build();

    sesClient.deleteEmailTemplate(deleteTemplateRequest);
```

```
        System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
        // If the email template does not exist, log the error and proceed
        System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
        System.err.println("Error occurred while deleting the email template: " +
            e.getMessage());
        e.printStackTrace();
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateContact](#)
 - [CreateContactList](#)
 - [CreateEmailIdentity](#)
 - [CreateEmailTemplate](#)
 - [DeleteContactList](#)
 - [DeleteEmailIdentity](#)
 - [DeleteEmailTemplate](#)
 - [ListContacts](#)
 - [SendEmail.simple](#)
 - [SendEmail.modèle](#)

Exemples Amazon SNS utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon SNS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Hello Amazon SNS

Les exemples de code suivants montrent comment démarrer avec Amazon SNS.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                .flatMap(r -> r.topics().stream())
                .forEach(content -> System.out.println(" Topic ARN: " +
                    content.topicArn()));
        } catch (SnsException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTopics](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)
- [Exemples sans serveur](#)

Actions

CheckIfPhoneNumberIsOptedOut

L'exemple de code suivant montre comment utiliser `CheckIfPhoneNumberIsOptedOut`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <phoneNumber>

            Where:
                phoneNumber - The mobile phone number to look up (for example,
+1XXX5550100).

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
CheckIfPhoneNumberIsOptedOutRequest.builder()
                .phoneNumber(phoneNumber)
                .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
```

```
        result.isOptedOut() + "Phone Number " + phoneNumber + " has
Opted Out of receiving sns messages." +
        "\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CheckIfPhoneNumberIsOptedOut](#) à la section Référence des AWS SDK for Java 2.x API.

ConfirmSubscription

L'exemple de code suivant montre comment utiliser `ConfirmSubscription`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
*/
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionToken> <topicArn>

            Where:
                subscriptionToken - A short-lived token sent to an endpoint
during the Subscribe action.
                topicArn - The ARN of the topic.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
ConfirmSubscriptionRequest.builder()
                .token(subscriptionToken)
                .topicArn(topicArn)
                .build();

            ConfirmSubscriptionResponse result =
snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                + result.subscriptionArn());
        } catch (SnsException e) {
```



```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ConfirmSubscription](#) à la section Référence des AWS SDK for Java 2.x API.

CreateTopic

L'exemple de code suivant montre comment utiliser `CreateTopic`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

        Usage:    <topicName>
```

```
        Where:
            topicName - The name of the topic to create (for example,
mytopic).

        "";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String topicName = args[0];
    System.out.println("Creating a topic with name: " + topicName);
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String arnVal = createSNSTopic(snsClient, topicName);
    System.out.println("The topic ARN is" + arnVal);
    snsClient.close();
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTopic](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteTopic

L'exemple de code suivant montre comment utiliser `DeleteTopic`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <topicArn>

                Where:
                    topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String topicArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println("Deleting a topic with name: " + topicArn);
    deleteSNSTopic(snsClient, topicArn);
    snsClient.close();
}

public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTopic](#) à la section Référence des AWS SDK for Java 2.x API.

GetSMSAttributes

L'exemple de code suivant montre comment utiliser `GetSMSAttributes`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSMSAttributes {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn>

                Where:
                    topicArn - The ARN of the topic from which to retrieve
attributes.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    getSNSAttrutes(snsClient, topicArn);
    snsClient.close();
}

public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
    try {
        GetSubscriptionAttributesRequest request =
        GetSubscriptionAttributesRequest.builder()
            .subscriptionArn(topicArn)
            .build();

        // Get the Subscription attributes
        GetSubscriptionAttributesResponse res =
        snsClient.getSubscriptionAttributes(request);
        Map<String, String> map = res.attributes();

        // Iterate through the map
        Iterator iter = map.entrySet().iterator();
        while (iter.hasNext()) {
            Map.Entry entry = (Map.Entry) iter.next();
            System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
            entry.getValue());
        }

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("\n\nStatus was good");
}
}
```

- Pour de plus amples informations sur l'API, consultez [GetSMSAttributes](#) dans Référence de l'API AWS SDK for Java 2.x .

GetTopicAttributes

L'exemple de code suivant montre comment utiliser `GetTopicAttributes`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn>

                Where:
                    topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        System.out.println("Getting attributes for a topic with name: " + topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
                .topicArn(topicArn)
                .build();

            GetTopicAttributesResponse result =
snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetTopicAttributes](#) à la section Référence des AWS SDK for Java 2.x API.

ListPhoneNumbersOptedOut

L'exemple de code suivant montre comment utiliser `ListPhoneNumbersOptedOut`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
```



```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " + result.sdkHttpResponse().statusCode())
+ "\n\nPhone Numbers: \n\n"
                + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListPhoneNumbersOptedOut](#) à la section Référence des AWS SDK for Java 2.x API.

ListSubscriptions

L'exemple de code suivant montre comment utiliser `ListSubscriptions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                .build();

            ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());
        }
    }
}
```

```
        } catch (SnsException e) {  
  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [ListSubscriptions](#) à la section Référence des AWS SDK for Java 2.x API.

ListTopics

L'exemple de code suivant montre comment utiliser `ListTopics`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;  
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ListTopics {  
    public static void main(String[] args) {
```

```
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

listSNSTopics(snsClient);
snsClient.close();
}

public static void listSNSTopics(SnsClient snsClient) {
    try {
        ListTopicsRequest request = ListTopicsRequest.builder()
            .build();

        ListTopicsResponse result = snsClient.listTopics(request);
        System.out.println(
            "Status was " + result.sdkHttpResponse().statusCode() + "\n\n"
            + result.topics());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTopics](#) à la section Référence des AWS SDK for Java 2.x API.

Publish

L'exemple de code suivant montre comment utiliser Publish.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <topicArn>

            Where:
                message - The message text to send.
                topicArn - The ARN of the topic to publish.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
```

```

        .message(message)
        .topicArn(topicArn)
        .build();

    PublishResponse result = snsClient.publish(request);
    System.out
        .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Pour de plus amples informations sur l'API, consultez [Publier](#) dans Référence de l'API AWS SDK for Java 2.x .

SetSMSAttributes

L'exemple de code suivant montre comment utiliser `SetSMSAttributes`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development

```

```

* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSMSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSMSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Pour les détails d'API, consultez [SetSMSAttributes](#) dans Référence de l'API AWS SDK for Java 2.x .

SetSubscriptionAttributes

L'exemple de code suivant montre comment utiliser `SetSubscriptionAttributes`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <subscriptionArn>

                Where:
                    subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```



```
        .region(Region.US_EAST_1)
        .build();

    usePolicy(snsClient, subscriptionArn);
    snsClient.close();
}

public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
    try {
        SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
        // Add a filter policy attribute with a single value
        fp.addAttribute("store", "example_corp");
        fp.addAttribute("event", "order_placed");

        // Add a prefix attribute
        fp.addAttributePrefix("customer_interests", "bas");

        // Add an anything-but attribute
        fp.addAttributeAnythingBut("customer_interests", "baseball");

        // Add a filter policy attribute with a list of values
        ArrayList<String> attributeValues = new ArrayList<>();
        attributeValues.add("rugby");
        attributeValues.add("soccer");
        attributeValues.add("hockey");
        fp.addAttribute("customer_interests", attributeValues);

        // Add a numeric attribute
        fp.addAttribute("price_usd", "=", 0);

        // Add a numeric attribute with a range
        fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

        // Apply the filter policy attributes to an Amazon SNS subscription
        fp.apply(snsClient, subscriptionArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [SetSubscriptionAttributes](#) à la section Référence des AWS SDK for Java 2.x API.

SetTopicAttributes

L'exemple de code suivant montre comment utiliser `SetTopicAttributes`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <attribute> <topicArn> <value>

            Where:
                attribute - The attribute action to use. Valid parameters are:
                Policy | DisplayName | DeliveryPolicy .
                topicArn - The ARN of the topic.\s
                value - The value for the attribute.

            """;
```

```
        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {
        try {
            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                .attributeName(attribute)
                .attributeValue(value)
                .topicArn(topicArn)
                .build();

            SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
            System.out.println(
                "\n\nStatus was " + result.sdkHttpResponse().statusCode() + "\n
\nTopic " + request.topicArn()
                    + " updated " + request.attributeName() + " to " +
request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SetTopicAttributes](#) à la section Référence des AWS SDK for Java 2.x API.

Subscribe

L'exemple de code suivant montre comment utiliser `Subscribe`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Abonnez une adresse e-mail à un sujet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <topicArn> <email>

            Where:
                topicArn - The ARN of the topic to subscribe.
                email - The email address to use.
            """;

        if (args.length != 2) {
```

```

        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    String email = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    subEmail(snsClient, topicArn, email);
    snsClient.close();
}

public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

Abonnez un point de terminaison HTTP à une rubrique.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;

```

```
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn> <url>

            Where:
                topicArn - The ARN of the topic to subscribe.
                url      - The HTTPS endpoint that you want to receive notifications.
            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("https")
                .endpoint(url)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();
        }
    }
}
```

```
        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN is " + result.subscriptionArn() +
"\n\n Status is "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Abonne une fonction Lambda à une rubrique.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <topicArn> <lambdaArn>

            Where:
                topicArn - The ARN of the topic to subscribe.
                lambdaArn - The ARN of an AWS Lambda function.
            """;
    }
}
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("lambda")
                .endpoint(lambdaArn)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Pour de plus amples informations sur l'API, consultez [S'abonner](#) dans Référence de l'API AWS SDK for Java 2.x .

TagResource

L'exemple de code suivant montre comment utiliser `TagResource`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn>

                Where:
                    topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String topicArn = args[0];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

addTopicTags(snsClient, topicArn);
snsClient.close();
}

public static void addTopicTags(SnsClient snsClient, String topicArn) {
    try {
        Tag tag = Tag.builder()
            .key("Team")
            .value("Development")
            .build();

        Tag tag2 = Tag.builder()
            .key("Environment")
            .value("Gamma")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag);
        tagList.add(tag2);

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(topicArn)
            .tags(tagList)
            .build();

        snsClient.tagResource(tagResourceRequest);
        System.out.println("Tags have been added to " + topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [TagResource](#) à la section Référence des AWS SDK for Java 2.x API.

Unsubscribe

L'exemple de code suivant montre comment utiliser `Unsubscribe`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <subscriptionArn>

                Where:
                    subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String subscriptionArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    unSub(snsClient, subscriptionArn);
    snsClient.close();
}

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
            + "\n\nSubscription was removed for " +
request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour de plus amples informations sur l'API, consultez [Se désabonner](#) dans Référence de l'API AWS SDK for Java 2.x .

Scénarios

Créer un point de terminaison de plateforme pour les notifications push

L'exemple de code suivant montre comment créer un point de terminaison de plateforme pour les notifications push Amazon SNS.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <token> <platformApplicationArn>

                Where:
                    token - The name of the FIFO topic.\s
                    platformApplicationArn - The ARN value of platform application.
                You can get this value from the AWS Management Console.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Créer et publier dans une rubrique FIFO

L'exemple de code suivant montre comment créer et publier dans une rubrique FIFO Amazon SNS.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Cet exemple

- crée une rubrique FIFO Amazon SNS, deux files d'attente FIFO Amazon SQS et une file d'attente standard.
- abonne les files d'attente à la rubrique et publie un message dans la rubrique.

Le [test](#) vérifie que le message a bien été reçu pour chaque file d'attente. L'[exemple complet](#) montre également l'ajout de stratégies d'accès et supprime les ressources à la fin.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
```

```
final String wholeSaleQueueName = args[1];
final String retailQueueName = args[2];
final String analyticsQueueName = args[3];

// For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
// name and type.
List<QueueData> queues = List.of(
    new QueueData(wholeSaleQueueName, QueueType.FIFO),
    new QueueData(retailQueueName, QueueType.FIFO),
    new QueueData(analyticsQueueName, QueueType.Standard));

// Create queues.
createQueues(queues);

// Create a topic.
String topicARN = createFIFOTopic(fifoTopicName);

// Subscribe each queue to the topic.
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
```



```
        .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {
    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";
```

```
MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
    .dataType("String")
    .stringValue(attributeValue)
    .build();

Map<String, MessageAttributeValue> attributes = new HashMap<>();
attributes.put(attributeName, msgAttValue);
PublishRequest pubRequest = PublishRequest.builder()
    .topicArn(topicArn)
    .subject(subject)
    .message(payload)
    .messageGroupId(groupId)
    .messageDeduplicationId(dedupId)
    .messageAttributes(attributes)
    .build();

final PublishResponse response = snsClient.publish(pubRequest);
System.out.println(response.messageId());
System.out.println(response.sequenceNumber());
System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Publish](#)
 - [Subscribe](#)

Publier des messages SMS dans une rubrique

L'exemple de code suivant illustre comment :

- Créer une rubrique Amazon SNS.
- Abonner des numéros de téléphone à la rubrique.

- Publier des messages SMS dans la rubrique afin que tous les numéros de téléphone abonnés reçoivent le message en même temps.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créer une rubrique et renvoyez son ARN.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicName>

                Where:
                    topicName - The name of the topic to create (for example,
mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

    }

    String topicName = args[0];
    System.out.println("Creating a topic with name: " + topicName);
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String arnVal = createSNSTopic(snsClient, topicName);
    System.out.println("The topic ARN is" + arnVal);
    snsClient.close();
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
}

```

Abonner un point de terminaison à une rubrique

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn> <phoneNumber>

            Where:
                topicArn - The ARN of the topic to subscribe.
                phoneNumber - A mobile phone number that receives notifications
(for example, +1XXX5550100).
            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("sms")
                .endpoint(phoneNumber)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
```

```

        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
        + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

Définir des attributs sur le message, tels que l'ID de l'expéditeur, le prix maximal et son type. Les attributs de message sont facultatifs.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }
}

```

```

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Publier un message dans une rubrique Le message est envoyé à chaque abonné.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

            Where:

```

```
        message - The message text to send.
        phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String phoneNumber = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();
    pubTextSMS(snsClient, message, phoneNumber);
    snsClient.close();
}

public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Publier un message texte SMS

L'exemple de code suivant montre comment publier des SMS à l'aide d'Amazon SNS.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <message> <phoneNumber>

                Where:
                    message - The message text to send.
                    phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();
    pubTextSMS(snsClient, message, phoneNumber);
    snsClient.close();
}

public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour de plus amples informations sur l'API, consultez [Publier](#) dans Référence de l'API AWS SDK for Java 2.x .

Exemples sans serveur

Invocation d'une fonction lambda à partir d'un déclencheur Amazon SNS

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages provenant d'une rubrique SNS. La fonction extrait les messages du paramètre d'événement et consigne le contenu de chaque message.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement SNS avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
        }
    }
}
```

```
        logger.log("message: " + message);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
}
```

Exemples Amazon SQS utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon SQS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Amazon SQS

Les exemples de code suivants montrent comment commencer à utiliser Amazon SQS.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                .flatMap(r -> r.queueUrls().stream())
                .forEach(content -> System.out.println(" Queue URL: " +
content.toLowerCase()));

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListQueues](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)
- [Exemples sans serveur](#)

Actions

CreateQueue

L'exemple de code suivant montre comment utiliser `CreateQueue`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient
                .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

```
}

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
        ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
    // List queues with filters
    String namePrefix = "queue";
    ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
        .queueNamePrefix(namePrefix)
        .build();

    ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
    System.out.println("Queue URLs with prefix: " + namePrefix);
    for (String url : listQueuesFilteredResponse.queueUrls()) {
        System.out.println(url);
    }

    System.out.println("\nSend message");
    try {
        sqsClient.sendMessage(SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody("Hello world!")
            .delaySeconds(10)
            .build());
    }
```



```
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nSend multiple messages");
    try {
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
            .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)

                .build())
            .build();
        sqsClient.sendMessageBatch(sendMessageBatchRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

    System.out.println("\nReceive messages");
    try {
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
            .queueUrl(queueUrl)
            .numberOfMessages(5)
            .build();
        return sqsClient.receiveMessage(receiveMessageRequest).messages();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }
    return null;
}

public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nChange Message Visibility");
    try {

        for (Message message : messages) {
            ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .visibilityTimeout(100)
                .build();
            sqsClient.changeMessageVisibility(req);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    System.out.println("\nDelete Messages");

    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateQueue](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteMessage

L'exemple de code suivant montre comment utiliser `DeleteMessage`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteMessage](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteQueue

L'exemple de code suivant montre comment utiliser `DeleteQueue`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <queueName>

            Where:
                queueName - The name of the Amazon SQS queue to delete.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        SqsClient sqs = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();
```

```
        deleteSQSQueue(sqs, queueName);
        sqs.close();
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteQueue](#) à la section Référence des AWS SDK for Java 2.x API.

GetQueueUrl

L'exemple de code suivant montre comment utiliser `GetQueueUrl`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
GetQueueUrlResponse getQueueUrlResponse = sqsClient
    .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
```

- Pour plus de détails sur l'API, reportez-vous [GetQueueUrl](#) à la section Référence des AWS SDK for Java 2.x API.

ListQueues

L'exemple de code suivant montre comment utiliser `ListQueues`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- Pour plus de détails sur l'API, reportez-vous [ListQueues](#) à la section Référence des AWS SDK for Java 2.x API.

ReceiveMessage

L'exemple de code suivant montre comment utiliser `ReceiveMessage`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .numberOfMessages(5)
        .build();
    return sqsClient.receiveMessage(receiveMessageRequest).messages();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

- Pour plus de détails sur l'API, reportez-vous [ReceiveMessage](#) à la section Référence des AWS SDK for Java 2.x API.

SendMessage

L'exemple de code suivant montre comment utiliser `SendMessage`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <queueName> <message>

                Where:
                    queueName - The name of the queue.
                    message - The message to send.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }

    public static void sendMessage(SqsClient sqsClient, String queueName, String
message) {
```



```
try {
    CreateQueueRequest request = CreateQueueRequest.builder()
        .queueName(queueName)
        .build();
    sqsClient.createQueue(request);

    GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
        .queueName(queueName)
        .build();

    String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
    SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
        .queueUrl(queueUrl)
        .messageBody(message)
        .delaySeconds(5)
        .build();

    sqsClient.sendMessage(sendMsgRequest);

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [SendMessage](#) à la section Référence des AWS SDK for Java 2.x API.

SendMessageBatch

L'exemple de code suivant montre comment utiliser `SendMessageBatch`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)

        .build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Pour plus de détails sur l'API, reportez-vous [SendMessageBatch](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Créer et publier dans une rubrique FIFO

L'exemple de code suivant montre comment créer et publier dans une rubrique FIFO Amazon SNS.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Cet exemple

- crée une rubrique FIFO Amazon SNS, deux files d'attente FIFO Amazon SQS et une file d'attente standard.
- abonne les files d'attente à la rubrique et publie un message dans la rubrique.

Le [test](#) vérifie que le message a bien été reçu pour chaque file d'attente. L'[exemple complet](#) montre également l'ajout de stratégies d'accès et supprime les ressources à la fin.

```
public class PriceUpdateExample {
```

```
public final static SnsClient snsClient = SnsClient.create();
public final static SqsClient sqsClient = SqsClient.create();

public static void main(String[] args) {

    final String usage = "\n" +
        "Usage: " +
        "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
        "Where:\n" +
        "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
        "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
        +
        "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
        "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    final String fifoTopicName = args[0];
    final String wholeSaleQueueName = args[1];
    final String retailQueueName = args[2];
    final String analyticsQueueName = args[3];

    // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
    // name and type.
    List<QueueData> queues = List.of(
        new QueueData(wholeSaleQueueName, QueueType.FIFO),
        new QueueData(retailQueueName, QueueType.FIFO),
        new QueueData(analyticsQueueName, QueueType.Standard));

    // Create queues.
    createQueues(queues);

    // Create a topic.
    String topicARN = createFIFOTopic(fifoTopicName);

    // Subscribe each queue to the topic.
```

```
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOtopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
```

```
        .endpoint(queue.queueARN)
        .protocol("sqs")
        .build();

    // Subscribe to the endpoint by using the SNS service client.
    // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
    // topic.
    SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
    System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
    queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";

        MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
            .dataType("String")
            .stringValue(attributeValue)
            .build();

        Map<String, MessageAttributeValue> attributes = new HashMap<>();
        attributes.put(attributeName, msgAttValue);
        PublishRequest pubRequest = PublishRequest.builder()
            .topicArn(topicArn)
            .subject(subject)
            .message(payload)
            .messageGroupId(groupId)
            .messageDeduplicationId(dedupId)
            .messageAttributes(attributes)
            .build();

        final PublishResponse response = snsClient.publish(pubRequest);
        System.out.println(response.messageId());
    }
}
```

```
        System.out.println(response.sequenceNumber());
        System.out.println("Message was published to " + topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Publish](#)
 - [Subscribe](#)

Exemples sans serveur

Invoquer une fonction Lambda à partir d'un déclencheur Amazon SQS

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages provenant d'une file d'attente SQS. La fonction extrait les messages du paramètre d'événement et consigne le contenu de chaque message.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement SQS avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur Amazon SQS

L'exemple de code suivant montre comment implémenter une réponse par lots partielle pour les fonctions Lambda qui reçoivent des événements d'une file d'attente SQS. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Signalement des échecs d'articles par lots SQS avec Lambda à l'aide de Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

Exemples de Step Functions utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK for Java 2.x with Step Functions.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Step Functions

Les exemples de code suivants montrent comment commencer à utiliser Step Functions.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Version Java de Hello.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
```

```
public static void main(String[] args) {
    Region region = Region.US_EAST_1;
    SfnClient sfnClient = SfnClient.builder()
        .region(region)
        .build();

    listMachines(sfnClient);
    sfnClient.close();
}

public static void listMachines(SfnClient sfnClient) {
    try {
        ListStateMachinesResponse response = sfnClient.listStateMachines();
        List<StateMachineListItem> machines = response.stateMachines();
        for (StateMachineListItem machine : machines) {
            System.out.println("The name of the state machine is: " +
machine.name());
            System.out.println("The ARN value is : " +
machine.stateMachineArn());
        }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListStateMachines](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateActivity

L'exemple de code suivant montre comment utiliser `CreateActivity`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();


    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateActivity](#) à la section Référence des AWS SDK for Java 2.x API.

CreateStateMachine

L'exemple de code suivant montre comment utiliser `CreateStateMachine`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateStateMachine](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteActivity

L'exemple de code suivant montre comment utiliser `DeleteActivity`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
```

```
        .activityArn(actArn)
        .build();

    sfnClient.deleteActivity(activityRequest);
    System.out.println("You have deleted " + actArn);

} catch (SfnException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteActivity](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteStateMachine

L'exemple de code suivant montre comment utiliser `DeleteStateMachine`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();
```

```
        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteStateMachine](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeExecution

L'exemple de code suivant montre comment utiliser `DescribeExecution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
```

```
DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
    status = response.statusAsString();
    if (status.compareTo("RUNNING") == 0) {
        System.out.println("The state machine is still running, let's
wait for it to finish.");
        Thread.sleep(2000);
    } else if (status.compareTo("SUCCEEDED") == 0) {
        System.out.println("The Step Function workflow has succeeded");
        hasSucceeded = true;
    } else {
        System.out.println("The Status is neither running or
succeeded");
    }
    System.out.println("The Status is " + status);

} catch (SfnException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeExecution](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeStateMachine

L'exemple de code suivant montre comment utiliser `DescribeStateMachine`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
```

```
try {
    DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
        .stateMachineArn(stateMachineArn)
        .build();

    DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
    System.out.println("The name of the State machine is " +
response.name());
    System.out.println("The status of the State machine is " +
response.status());
    System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
    System.out.println("The role ARN value is " + response.roleArn());

} catch (SfnException e) {
    System.err.println(e.getMessage());
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeStateMachine](#) à la section Référence des AWS SDK for Java 2.x API.

GetActivityTask

L'exemple de code suivant montre comment utiliser `GetActivityTask`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
```



```
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

/// <summary>
/// Stop execution of a Step Functions workflow.
/// </summary>
/// <param name="executionArn">The Amazon Resource Name (ARN) of
/// the Step Functions execution to stop.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StopExecution(string executionArn)
{
    var response =
        await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
{ ExecutionArn = executionArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetActivityTask](#) à la section Référence des AWS SDK for Java 2.x API.

ListActivities

L'exemple de code suivant montre comment utiliser `ListActivities`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
ListActivitiesRequest.builder()
                .maxResults(10)
                .build();

            ListActivitiesResponse response =
sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }
        } catch (SfnException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListActivities](#) à la section Référence des AWS SDK for Java 2.x API.

ListExecutions

L'exemple de code suivant montre comment utiliser `ListExecutions`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
    try {
        GetExecutionHistoryRequest historyRequest =
        GetExecutionHistoryRequest.builder()
            .executionArn(exeARN)
            .maxResults(10)
            .build();

        GetExecutionHistoryResponse historyResponse =
        sfnClient.getExecutionHistory(historyRequest);
        List<HistoryEvent> events = historyResponse.events();
        for (HistoryEvent event : events) {
            System.out.println("The event type is " + event.type().toString());
        }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [ListExecutions](#) à la section Référence des AWS SDK for Java 2.x API.

ListStateMachines

L'exemple de code suivant montre comment utiliser `ListStateMachines`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
    }
}
```

```
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
                    machine.name());
                System.out.println("The ARN value is : " +
                    machine.stateMachineArn());
            }
        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListStateMachines](#) à la section Référence des AWS SDK for Java 2.x API.

SendTaskSuccess

L'exemple de code suivant montre comment utiliser `SendTaskSuccess`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
```

```
        .taskToken(token)
        .output(json)
        .build();

    sfnClient.sendTaskSuccess(successRequest);

} catch (SfnException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [SendTaskSuccess](#) à la section Référence des AWS SDK for Java 2.x API.

StartExecution

L'exemple de code suivant montre comment utiliser `StartExecution`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
            sfnClient.startExecution(executionRequest);
    }
```

```
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [StartExecution](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Commencez avec les machines d'État

L'exemple de code suivant illustre comment :

- Créez une activité.
- Créez une machine à états à partir d'une définition du langage Amazon States qui contient l'activité créée précédemment en tant qu'étape.
- Exécutez la machine d'état et répondez à l'activité avec la saisie de l'utilisateur.
- Obtenez le statut final et le résultat une fois l'exécution terminée, puis nettoyez les ressources.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 */
```

```
* To run this code example, place the chat_sfn_state_machine.json file into
* your project's resources folder.
*
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java code example performs the following tasks:
*
* 1. Creates an activity.
* 2. Creates a state machine.
* 3. Describes the state machine.
* 4. Starts execution of the state machine and interacts with it.
* 5. Describes the execution.
* 6. Delete the activity.
* 7. Deletes the state machine.
*/
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <roleARN> <activityName> <stateMachineName>

            Where:
                roleName - The name of the IAM role to create for this state
machine.

                activityName - The name of an activity to create.
                stateMachineName - The name of the state machine to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String polJSON = "{\n" +
```



```

    "    \"Version\": \"2012-10-17\", \"n\" +
    "    \"Statement\": [\"n\" +
    "        {\"n\" +
    "            \"Sid\": \"\", \"n\" +
    "            \"Effect\": \"Allow\", \"n\" +
    "            \"Principal\": {\"n\" +
    "                \"Service\": \"states.amazonaws.com\", \"n\" +
    "            }, \"n\" +
    "            \"Action\": \"sts:AssumeRole\", \"n\" +
    "        } \"n\" +
    "    ] \"n\" +
    "};

```

```

Scanner sc = new Scanner(System.in);
boolean action = false;

```

```

Region region = Region.US_EAST_1;
SfnClient sfnClient = SfnClient.builder()
    .region(region)
    .build();

```

```

Region regionGl = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(regionGl)
    .build();

```

```

System.out.println(DASHES);
System.out.println("Welcome to the AWS Step Functions example scenario.");
System.out.println(DASHES);

```

```

System.out.println(DASHES);
System.out.println("1. Create an activity.");
String activityArn = createActivity(sfnClient, activityName);
System.out.println("The ARN of the activity is " + activityArn);
System.out.println(DASHES);

```

```

// Get JSON to use for the state machine and place the activityArn value
into

```

```

// it.
InputStream input = StepFunctionsScenario.class.getClassLoader()
    .getResourceAsStream("chat_sfn_state_machine.json");
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
String jsonString = mapper.writeValueAsString(jsonNode);

```

```
// Modify the Resource node.
ObjectMapper objectMapper = new ObjectMapper();
JsonNode root = objectMapper.readTree(jsonString);
((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

// Convert the modified Java object back to a JSON string.
String stateDefinition = objectMapper.writeValueAsString(root);
System.out.println(stateDefinition);

System.out.println(DASHES);
System.out.println("2. Create a state machine.");
String roleARN = createIAMRole(iam, roleName, polJSON);
String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
System.out.println("The ARN of the state machine is " + stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe the state machine.");
describeStateMachine(sfnClient, stateMachineArn);
System.out.println("What should ChatSFN call you?");
String userName = sc.nextLine();
System.out.println("Hello " + userName);
System.out.println(DASHES);

System.out.println(DASHES);
// The JSON to pass to the StartExecution call.
String executionJson = "{ \"name\" : \"" + userName + "\" }";
System.out.println(executionJson);
System.out.println("4. Start execution of the state machine and interact
with it.");
String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
System.out.println("The ARN of the state machine execution is " + runArn);
List<String> myList;
while (!action) {
    myList = getActivityTask(sfnClient, activityArn);
    System.out.println("ChatSFN: " + myList.get(1));
    System.out.println(userName + " please specify a value.");
    String myAction = sc.nextLine();
    if (myAction.compareTo("done") == 0)
        action = true;
}
```

```
        System.out.println("You have selected " + myAction);
        String taskJson = "{ \"action\" : \"" + myAction + "\" }";
        System.out.println(taskJson);
        sendTaskSuccess(sfnClient, myList.get(0), taskJson);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Describe the execution.");
    describeExe(sfnClient, runArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Delete the activity.");
    deleteActivity(sfnClient, activityArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Delete the state machines.");
    deleteMachine(sfnClient, stateMachineArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The AWS Step Functions example scenario is complete.");
    System.out.println(DASHES);
}

public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}

public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
            status = response.statusAsString();
            if (status.compareTo("RUNNING") == 0) {
                System.out.println("The state machine is still running, let's
wait for it to finish.");
                Thread.sleep(2000);
            } else if (status.compareTo("SUCCEEDED") == 0) {
                System.out.println("The Step Function workflow has succeeded");
                hasSucceeded = true;
            } else {
                System.out.println("The Status is neither running or
succeeded");
            }
        }
        System.out.println("The Status is " + status);

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);
    }
}
```

```
    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> getActivityTask(SfnClient sfncClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
sfncClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

public static void deleteActivity(SfnClient sfncClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();

        sfncClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeStateMachine(SfnClient sfncClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();
```

```
        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());

    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}

public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}

public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
```

```
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                .input(jsonEx)
                .stateMachineArn(stateMachineArn)
                .name(uuidValue)
                .build();

            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
                .definition(json)
                .name(stateMachineName)
                .roleArn(roleARN)
                .type(StateMachineType.STANDARD)
                .build();

            CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
```

```
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
            sfmClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CreateActivity](#)
 - [CreateStateMachine](#)
 - [DeleteActivity](#)
 - [DeleteStateMachine](#)
 - [DescribeExecution](#)
 - [DescribeStateMachine](#)
 - [GetActivityTask](#)
 - [ListActivities](#)
 - [ListStateMachines](#)
 - [SendTaskSuccess](#)
 - [StartExecution](#)
 - [StopExecution](#)

AWS STS exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with AWS STS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

AssumeRole

L'exemple de code suivant montre comment utiliser `AssumeRole`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
```

```

* To make this code example work, create a Role that you want to assume.
* Then define a Trust Relationship in the AWS Console. You can use this as an
* example:
*
* {
* "Version": "2012-10-17",
* "Statement": [
* {
* "Effect": "Allow",
* "Principal": {
* "AWS": "<Specify the ARN of your IAM user you are using in this code
* example>"
* },
* "Action": "sts:AssumeRole"
* }
* ]
* }
*
* For more information, see "Editing the Trust Relationship for an Existing
* Role" in the AWS Directory Service guide.
*
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleArn> <roleSessionName>\s

            Where:
                roleArn - The Amazon Resource Name (ARN) of the role to assume
(for example, rn:aws:iam::000008047983:role/s3role).\s
                roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);

```

```
    }

    String roleArn = args[0];
    String roleSessionName = args[1];
    Region region = Region.US_EAST_1;
    StsClient stsClient = StsClient.builder()
        .region(region)
        .build();

    assumeGivenRole(stsClient, roleArn, roleSessionName);
    stsClient.close();
}

public static void assumeGivenRole(StsClient stsClient, String roleArn, String
roleSessionName) {
    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();

        // Display the time when the temp creds expire.
        Instant exTime = myCreds.expiration();
        String tokenInfo = myCreds.sessionToken();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(exTime);
        System.out.println("The token " + tokenInfo + " expires on " + exTime);

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [AssumeRole](#) à la section Référence des AWS SDK for Java 2.x API.

AWS Support exemples d'utilisation du SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK for Java 2.x with AWS Support.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour AWS Support

Les exemples de code suivants montrent comment démarrer avec AWS Support.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
```

```
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                .language("en")
                .build();
```

```
DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
List<Service> services = response.services();

System.out.println("Get the first 10 services");
int index = 1;
for (Service service : services) {
    if (index == 11)
        break;

    System.out.println("The Service name is: " + service.name());

    // Display the Categories for this service.
    List<Category> categories = service.categories();
    for (Category cat : categories) {
        System.out.println("The category name is: " + cat.name());
    }
    index++;
}

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeServices](#) à la section Référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

AddAttachmentsToSet

L'exemple de code suivant montre comment utiliser `AddAttachmentsToSet`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
            .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [AddAttachmentsToSet](#) à la section Référence des AWS SDK for Java 2.x API.

AddCommunicationToCase

L'exemple de code suivant montre comment utiliser `AddCommunicationToCase`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [AddCommunicationToCase](#) à la section Référence des AWS SDK for Java 2.x API.

CreateCase

L'exemple de code suivant montre comment utiliser `CreateCase`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedName());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCase](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeAttachment

L'exemple de code suivant montre comment utiliser `DescribeAttachment`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());


    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAttachment](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeCases

L'exemple de code suivant montre comment utiliser `DescribeCases`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCases](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeCommunications

L'exemple de code suivant montre comment utiliser `DescribeCommunications`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm : communications) {
            System.out.println("the body is: " + comm.body());

            // Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCommunications](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeServices

L'exemple de code suivant montre comment utiliser `DescribeServices`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
                break;

            System.out.println("The Service name is: " + service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();
```

```
        // Get the Categories for this service.
        List<Category> categories = service.categories();
        for (Category cat : categories) {
            System.out.println("The category name is: " + cat.name());
            if (cat.name().compareTo("Security") == 0)
                catName = cat.name();
        }
        index++;
    }

    // Push the two values to the list.
    sevCatList.add(serviceCode);
    sevCatList.add(catName);
    return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeServices](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeSeverityLevels

L'exemple de code suivant montre comment utiliser `DescribeSeverityLevels`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String displaySevLevels(SupportClient supportClient) {
    try {
```

```
DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
    .language("en")
    .build();

DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
List<SeverityLevel> severityLevels = response.severityLevels();
String levelName = null;
for (SeverityLevel sevLevel : severityLevels) {
    System.out.println("The severity level name is: " +
sevLevel.name());
    if (sevLevel.name().compareTo("High") == 0)
        levelName = sevLevel.name();
}
return levelName;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSeverityLevels](#) à la section Référence des AWS SDK for Java 2.x API.

ResolveCase

L'exemple de code suivant montre comment utiliser `ResolveCase`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ResolveCase](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Démarrer avec les dossiers

L'exemple de code suivant illustre comment :

- Obtenez et affichez les services disponibles et les niveaux de gravité des dossiers.
- Créez un dossier de support en utilisant un service, une catégorie et un niveau de gravité sélectionnés.
- Obtenez et affichez une liste des dossiers ouverts pour la journée en cours.
- Ajoutez un ensemble de pièces jointes et une communication au nouveau dossier.
- Décrivez la nouvelle pièce jointe et la nouvelle communication relatives au dossier.
- Résolvez le dossier.
- Obtenez et affichez la liste des dossiers ouverts pour la journée en cours.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez diverses AWS Support opérations.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
```

```
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <fileAttachment>Where:
            fileAttachment - The file can be a simple saved .txt file to use
            as an email attachment.\s
            """;
    }
}
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String fileAttachment = args[0];
    Region region = Region.US_WEST_2;
    SupportClient supportClient = SupportClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("***** Welcome to the AWS Support case example
scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Get and display available services.");
    List<String> sevCatList = displayServices(supportClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get and display Support severity levels.");
    String sevLevel = displaySevLevels(supportClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Create a support case using the selected service,
category, and severity level.");
    String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
    if (caseId.compareTo("") == 0) {
        System.out.println("A support case was not successfully created!");
        System.exit(1);
    } else
        System.out.println("Support case " + caseId + " was successfully
created!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Get open support cases.");
    getOpenCase(supportClient);
    System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
```

```
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
        .maxResults(30)
        .afterTime(yesterday.toString())
        .beforeTime(now.toString())
        .includeResolvedCases(true)
        .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            if (sinCase.status().compareTo("resolved") == 0)
                System.out.println("The case status is " + sinCase.status());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeAttachment(SupportClient supportClient, String
attachId) {
```

```
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                .attachmentId(attachId)
                .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                .caseId(caseId)
                .maxResults(10)
                .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

```
    }
    return "";
}

public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
```

```
        .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
```



```
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

    } catch (SupportException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
                break;

            System.out.println("The Service name is: " + service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();

            // Get the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat : categories) {
                System.out.println("The category name is: " + cat.name());
                if (cat.name().compareTo("Security") == 0)
                    catName = cat.name();
            }
            index++;
        }

        // Push the two values to the list.
        sevCatList.add(serviceCode);
        sevCatList.add(catName);
    }
}
```

```
        return sevCatList;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return null;
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [AddAttachmentsToSet](#)
 - [AddCommunicationToCase](#)
 - [CreateCase](#)
 - [DescribeAttachment](#)
 - [DescribeCases](#)
 - [DescribeCommunications](#)
 - [DescribeServices](#)
 - [DescribeSeverityLevels](#)
 - [ResolveCase](#)

Exemples de Systems Manager utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK for Java 2.x with Systems Manager.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Hello Systems Manager

L'exemple de code suivant montre comment commencer à utiliser Systems Manager.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
              <awsAccount>

            Where:
              awsAccount - Your AWS Account number.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();
```

```
        listDocuments(ssmClient, awsAccount);
    }

    /**
     * This code automatically fetches the next set of results using the `nextToken`
     * and
     * stops once the desired maxResults (20 in this case) have been reached.
     */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
                System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- Pour plus de détails sur l'API, voir [ListThings](#) dans le manuel de référence des AWS SDK for Java 2.x API.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateDocument

L'exemple de code suivant montre comment utiliser `CreateDocument`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Create an AWS SSM document to use in this scenario.
public static void createSSMDoc(SsmClient ssmClient, String docName) {
    // Create JSON for the content
    String jsonData = ""
        {
            "schemaVersion": "2.2",
            "description": "Run a simple shell command",
            "mainSteps": [
                {
                    "action": "aws:runShellScript",
                    "name": "runEchoCommand",
                    "inputs": {
                        "runCommand": [
                            "echo 'Hello, world!'"
                        ]
                    }
                }
            ]
        }
        """;

    try {
        CreateDocumentRequest request = CreateDocumentRequest.builder()
            .content(jsonData)
            .name(docName)
            .documentType(DocumentType.COMMAND)
            .build();

        // Create the document.
```

```
        CreateDocumentResponse response = ssmClient.createDocument(request);
        System.out.println("The status of the document is " +
response.documentDescription().status());

        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The document already exists. Moving on." );
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDocument](#) à la section Référence des AWS SDK for Java 2.x API.

CreateMaintenanceWindow

L'exemple de code suivant montre comment utiliser `CreateMaintenanceWindow`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static String createMaintenanceWindow(SsmClient ssmClient, String
winName) {
    CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
        .name(winName)
        .description("This is my maintenance window")
        .allowUnassociatedTargets(true)
        .duration(2)
        .cutoff(1)
        .schedule("cron(0 10 ? * MON-FRI *)")
        .build();

    try {
```

```
        CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
        String maintenanceWindowId = response.windowId();
        System.out.println("The maintenance window id is " +
maintenanceWindowId);
        return maintenanceWindowId;

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The maintenance window already exists. Moving on.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
        .key("name")
        .values(winName)
        .build();

    DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
        .filters(filter)
        .build();

    String windowId = "";
    DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
    List<MaintenanceWindowIdentity> windows = response.windowIdentities();
    if (!windows.isEmpty()) {
        windowId = windows.get(0).windowId();
        System.out.println("Window ID: " + windowId);
    } else {
        System.out.println("Window not found.");
    }
    return windowId;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateMaintenanceWindow](#) à la section Référence des AWS SDK for Java 2.x API.

CreateOpsItem

L'exemple de code suivant montre comment utiliser `CreateOpsItem`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Create an SSM OpsItem
public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the Systems Manager Java API")
            .title(title)
            .source(source)
            .category(category)
            .severity(severity)
            .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
        return itemResponse.opsItemId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateOpsItem](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteDocument

L'exemple de code suivant montre comment utiliser `DeleteDocument`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Deletes an AWS Systems Manager document.
public static void deleteDoc(SsmClient ssmClient, String documentName) {
    try {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
            .name(documentName)
            .build();

        ssmClient.deleteDocument(documentRequest);
        System.out.println("The Systems Manager document was successfully
deleted.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDocument](#) à la section Référence des AWS SDK for Java 2.x API.

DeleteMaintenanceWindow

L'exemple de code suivant montre comment utiliser `DeleteMaintenanceWindow`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
    try {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
        .windowId(winId)
        .build();

        ssmClient.deleteMaintenanceWindow(windowRequest);
        System.out.println("The maintenance window was successfully deleted.");


    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteMaintenanceWindow](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeOpsItems

L'exemple de code suivant montre comment utiliser `DescribeOpsItems`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void describeOpsItems(SsmClient ssmClient, String key) {
    try {
        OpsItemFilter filter = OpsItemFilter.builder()
            .key(OpsItemFilterKey.OPS_ITEM_ID)
            .values(key)
            .operator(OpsItemFilterOperator.EQUAL)
            .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
            .maxResults(10)
            .opsItemFilters(filter)
            .build();

        DescribeOpsItemsResponse itemsResponse =
            ssmClient.describeOpsItems(itemsRequest);
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
        for (OpsItemSummary item : items) {
            System.out.println("The item title is " + item.title() + " and the
status is "+item.status().toString());
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeOpsItems](#) à la section Référence des AWS SDK for Java 2.x API.

DescribeParameters

L'exemple de code suivant montre comment utiliser `DescribeParameters`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetParameter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <paraName>

            Where:
                paraName - The name of the parameter.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
```

```
        .name(paraName)
        .build();

        GetParameterResponse parameterResponse =
            ssmClient.getParameter(parameterRequest);
        System.out.println("The parameter value is " +
            parameterResponse.parameter().value());

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeParameters](#) à la section Référence des AWS SDK for Java 2.x API.

PutParameter

L'exemple de code suivant montre comment utiliser `PutParameter`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = ""
```

```
        Usage:
            <paraName>

        Where:
            paraName - The name of the parameter.
            paraValue - The value of the parameter.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String paraName = args[0];
    String paraValue = args[1];
    Region region = Region.US_EAST_1;
    SsmClient ssmClient = SsmClient.builder()
        .region(region)
        .build();

    putParaValue(ssmClient, paraName, paraValue);
    ssmClient.close();
}

public static void putParaValue(SsmClient ssmClient, String paraName, String
value) {
    try {
        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(paraName)
            .type(ParameterType.STRING)
            .value(value)
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.println("The parameter was successfully added.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutParameter](#) à la section Référence des AWS SDK for Java 2.x API.

SendCommand

L'exemple de code suivant montre comment utiliser `SendCommand`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Sends a SSM command to a managed node.
public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
    // Before we use Document to send a command - make sure it is active.
    boolean isDocumentActive = false;
    DescribeDocumentRequest request = DescribeDocumentRequest.builder()
        .name(documentName)
        .build();

    while (!isDocumentActive) {
        DescribeDocumentResponse response = ssmClient.describeDocument(request);
        String documentStatus = response.document().statusAsString();
        if (documentStatus.equals("Active")) {
            System.out.println("The Systems Manager document is active and ready
to use.");
            isDocumentActive = true;
        } else {
            System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
            try {
                // Add a delay to avoid making too many requests.
                Thread.sleep(5000); // Wait for 5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```
    }
  }
}

// Create the SendCommandRequest.
SendCommandRequest commandRequest = SendCommandRequest.builder()
    .documentName(documentName)
    .instanceIds(instanceId)
    .build();

// Send the command.
SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
String commandId = commandResponse.command().commandId();
System.out.println("The command Id is " + commandId);

// Wait for the command execution to complete.
GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
    .commandId(commandId)
    .instanceId(instanceId)
    .build();

System.out.println("Wait 5 secs");
TimeUnit.SECONDS.sleep(5);

// Retrieve the command execution details.
GetCommandInvocationResponse commandInvocationResponse =
ssmClient.getCommandInvocation(invocationRequest);

// Check the status of the command execution.
CommandInvocationStatus status = commandInvocationResponse.status();
if (status == CommandInvocationStatus.SUCCESS) {
    System.out.println("Command execution successful.");
} else {
    System.out.println("Command execution failed. Status: " + status);
}
return commandId;
}
```

- Pour plus de détails sur l'API, reportez-vous [SendCommand](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateMaintenanceWindow

L'exemple de code suivant montre comment utiliser `UpdateMaintenanceWindow`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Update the maintenance window schedule
public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
String name) {
    try {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
            .windowId(id)
            .allowUnassociatedTargets(true)
            .duration(24)
            .enabled(true)
            .name(name)
            .schedule("cron(0 0 ? * MON *)")
            .build();

        ssmClient.updateMaintenanceWindow(updateRequest);
        System.out.println("The Systems Manager maintenance window was
successfully updated.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateMaintenanceWindow](#) à la section Référence des AWS SDK for Java 2.x API.

UpdateOpsItem

L'exemple de code suivant montre comment utiliser `UpdateOpsItem`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void resolveOpsItem(SsmClient ssmClient, String opsID) {
    try {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
            .opsItemId(opsID)
            .status(OpsItemStatus.RESOLVED)
            .build();

        ssmClient.updateOpsItem(opsItemRequest);

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateOpsItem](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Commencez avec Systems Manager

L'exemple de code suivant montre comment utiliser les fenêtres de maintenance de Systems Manager, les documents et OpsItems.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.CommandInvocation;
import software.amazon.awssdk.services.ssm.model.CommandInvocationStatus;
import software.amazon.awssdk.services.ssm.model.CreateDocumentRequest;
import software.amazon.awssdk.services.ssm.model.CreateDocumentResponse;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowResponse;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.DeleteDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DeleteMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.DeleteOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentResponse;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsResponse;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsResponse;
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.DocumentType;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationRequest;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationResponse;
import software.amazon.awssdk.services.ssm.model.GetOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.GetOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsRequest;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsResponse;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowFilter;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowIdentity;
import software.amazon.awssdk.services.ssm.model.OpsItemDataValue;
import software.amazon.awssdk.services.ssm.model.OpsItemFilter;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterKey;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterOperator;
import software.amazon.awssdk.services.ssm.model.OpsItemStatus;
```

```
import software.amazon.awssdk.services.ssm.model.OpsItemSummary;
import software.amazon.awssdk.services.ssm.model.SendCommandRequest;
import software.amazon.awssdk.services.ssm.model.SendCommandResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;
import software.amazon.awssdk.services.ssm.model.UpdateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.UpdateOpsItemRequest;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
 *
 * This Java program performs these tasks:
 * 1. Creates an AWS Systems Manager maintenance window with a default name or a
 * user-provided name.
 * 2. Modifies the maintenance window schedule.
 * 3. Creates a Systems Manager document with a default name or a user-provided
 * name.
 * 4. Sends a command to a specified EC2 instance using the created Systems Manager
 * document and displays the time when the command was invoked.
 * 5. Creates a Systems Manager OpsItem with a predefined title, source, category,
 * and severity.
 * 6. Updates and resolves the created OpsItem.
 * 7. Deletes the Systems Manager maintenance window, OpsItem, and document.
 */

public class SSMSscenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {
        String usage = ""
            Usage:
            <instanceId> <title> <source> <category> <severity>
    }
}
```

```
Where:
    instanceId - The Amazon EC2 Linux/UNIX instance Id that AWS Systems
Manager uses (ie, i-0149338494ed95f06).
    title - The title of the parameter (default is Disk Space Alert).
    source - The source of the parameter (default is EC2).
    category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
    severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

Scanner scanner = new Scanner(System.in);
String documentName;
String windowName;
String instanceId = args[0];
String title = "Disk Space Alert" ;
String source = "EC2" ;
String category = "Performance" ;
String severity = "2" ;

Region region = Region.US_EAST_1;
SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("""
    Welcome to the AWS Systems Manager SDK Getting Started scenario.
    This program demonstrates how to interact with Systems Manager using the
AWS SDK for Java (v2).
    Systems Manager is the operations hub for your AWS applications and
resources and a secure end-to-end management solution.
    The program's primary functions include creating a maintenance window,
creating a document, sending a command to a document,
    listing documents, listing commands, creating an OpsItem, modifying an
OpsItem, and deleting Systems Manager resources.
    Upon completion of the program, all AWS resources are cleaned up.
    Let's get started...
    """);
```

```
        Please hit Enter
        "");
    scanner.nextLine();
    System.out.println(DASHES);

    System.out.println("Create a Systems Manager maintenance window.");
    System.out.println("Please enter the maintenance window name (default is
    ssm-maintenance-window):");
    String win = scanner.nextLine();
    windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
    String winId = createMaintenanceWindow(ssmClient, windowName);
    System.out.println(DASHES);

    System.out.println("Modify the maintenance window by changing the
    schedule");
    System.out.println("Please hit Enter");
    scanner.nextLine();
    updateSSMMaintenanceWindow(ssmClient, winId, windowName);
    System.out.println(DASHES);

    System.out.println("Create a document that defines the actions that Systems
    Manager performs on your EC2 instance.");
    System.out.println("Please enter the document name (default is
    ssmdocument):");
    String doc = scanner.nextLine();
    documentName = doc.isEmpty() ? "ssmdocument" : doc;
    createSSMDoc(ssmClient, documentName);

    System.out.println("Now we are going to run a command on an EC2 instance
    that echoes 'Hello, world!'");
    System.out.println("Please hit Enter");
    scanner.nextLine();
    String commandId = sendSSMCommand(ssmClient, documentName, instanceId);
    System.out.println(DASHES);

    System.out.println("Lets get the time when the specific command was sent to
    the specific managed node");
    System.out.println("Please hit Enter");
    scanner.nextLine();
    displayCommands(ssmClient, commandId);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(""
```

Now we will create a Systems Manager OpsItem.

An OpsItem is a feature provided by the Systems Manager service.

It is a type of operational data item that allows you to manage and track various operational issues, events, or tasks within your AWS environment.

You can create OpsItems to track and manage operational issues as they arise.

For example, you could create an OpsItem whenever your application detects a critical error

or an anomaly in your infrastructure.

```
""");
```

```
System.out.println("Please hit Enter");
scanner.nextLine();
String opsItemId = createSSMOpsItem(ssmClient, title, source, category,
severity);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now we will update the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
String description = "An update to "+opsItemId ;
updateOpsItem(ssmClient, opsItemId, title, description);
System.out.println("Now we will get the status of the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
describeOpsItems(ssmClient, opsItemId);
System.out.println("Now we will resolve the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
resolveOpsItem(ssmClient, opsItemId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to delete the Systems Manager resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the resources.");
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    deleteOpsItem(ssmClient, opsItemId);
```



```
        deleteMaintenanceWindow(ssmClient, winId);
        deleteDoc(ssmClient, documentName);
    } else {
        System.out.println("The Systems Manager resources will not be deleted");
    }
    System.out.println(DASHES);

    System.out.println("This concludes the Systems Manager SDK Getting Started
scenario.");
    System.out.println(DASHES);
}

// Displays the date and time when the specific command was invoked.
public static void displayCommands(SsmClient ssmClient, String commandId) {
    try {
        ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
            .commandId(commandId)
            .build();

        ListCommandInvocationsResponse response =
ssmClient.listCommandInvocations(commandInvocationsRequest);
        List<CommandInvocation> commandList = response.commandInvocations();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
        for (CommandInvocation invocation : commandList) {
            System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Create an SSM OpsItem
public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the Systems Manager Java API")
            .title(title)
            .source(source)
```

```
        .category(category)
        .severity(severity)
        .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
        return itemResponse.opsItemId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Update the AWS SSM OpsItem.
public static void updateOpsItem(SsmClient ssmClient, String opsItemId, String
title, String description) {
    Map<String, OpsItemDataValue> operationalData = new HashMap<>();
    operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
    operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());

    try {
        UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
            .opsItemId(opsItemId)
            .title(title)
            .operationalData(operationalData)
            .status(getOpsItem(ssmClient, opsItemId))
            .description(description)
            .build();

        ssmClient.updateOpsItem(request);

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void resolveOpsItem(SsmClient ssmClient, String opsID) {
    try {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
```

```
        .opsItemId(opsID)
        .status(OpsItemStatus.RESOLVED)
        .build();

    ssmClient.updateOpsItem(opsItemRequest);

} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Gets a specific OpsItem.
private static OpsItemStatus getOpsItem(SsmClient ssmClient, String opsItemId) {
    GetOpsItemRequest itemRequest = GetOpsItemRequest.builder()
        .opsItemId(opsItemId)
        .build();

    try {
        GetOpsItemResponse response = ssmClient.getOpsItem(itemRequest);
        return response.opsItem().status();
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}

// Sends a SSM command to a managed node.
public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
    // Before we use Document to send a command - make sure it is active.
    boolean isDocumentActive = false;
    DescribeDocumentRequest request = DescribeDocumentRequest.builder()
        .name(documentName)
        .build();

    while (!isDocumentActive) {
        DescribeDocumentResponse response = ssmClient.describeDocument(request);
        String documentStatus = response.document().statusAsString();
        if (documentStatus.equals("Active")) {
            System.out.println("The Systems Manager document is active and ready
to use.");
        }
    }
}
```

```
        isDocumentActive = true;
    } else {
        System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
        try {
            // Add a delay to avoid making too many requests.
            Thread.sleep(5000); // Wait for 5 seconds before checking again
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

// Create the SendCommandRequest.
SendCommandRequest commandRequest = SendCommandRequest.builder()
    .documentName(documentName)
    .instanceIds(instanceId)
    .build();

// Send the command.
SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
String commandId = commandResponse.command().commandId();
System.out.println("The command Id is " + commandId);

// Wait for the command execution to complete.
GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
    .commandId(commandId)
    .instanceId(instanceId)
    .build();

System.out.println("Wait 5 secs");
TimeUnit.SECONDS.sleep(5);

// Retrieve the command execution details.
GetCommandInvocationResponse commandInvocationResponse =
ssmClient.getCommandInvocation(invocationRequest);

// Check the status of the command execution.
CommandInvocationStatus status = commandInvocationResponse.status();
if (status == CommandInvocationStatus.SUCCESS) {
    System.out.println("Command execution successful.");
} else {
    System.out.println("Command execution failed. Status: " + status);
}
```

```
    }
    return commandId;
}

// Deletes an AWS Systems Manager document.
public static void deleteDoc(SsmClient ssmClient, String documentName) {
    try {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
            .name(documentName)
            .build();

        ssmClient.deleteDocument(documentRequest);
        System.out.println("The Systems Manager document was successfully
deleted.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
    try {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
            .windowId(winId)
            .build();

        ssmClient.deleteMaintenanceWindow(windowRequest);
        System.out.println("The maintenance window was successfully deleted.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Update the maintenance window schedule
public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
String name) {
    try {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
            .windowId(id)
```

```
        .allowUnassociatedTargets(true)
        .duration(24)
        .enabled(true)
        .name(name)
        .schedule("cron(0 0 ? * MON *)")
        .build();

    ssmClient.updateMaintenanceWindow(updateRequest);
    System.out.println("The Systems Manager maintenance window was
successfully updated.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String createMaintenanceWindow(SsmClient ssmClient, String
winName) {
    CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
        .name(winName)
        .description("This is my maintenance window")
        .allowUnassociatedTargets(true)
        .duration(2)
        .cutoff(1)
        .schedule("cron(0 10 ? * MON-FRI *)")
        .build();

    try {
        CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
        String maintenanceWindowId = response.windowId();
        System.out.println("The maintenance window id is " +
maintenanceWindowId);
        return maintenanceWindowId;

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The maintenance window already exists. Moving on.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
    .key("name")
    .values(winName)
    .build();

DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
    .filters(filter)
    .build();

String windowId = "";
DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
List<MaintenanceWindowIdentity> windows = response.windowIdentities();
if (!windows.isEmpty()) {
    windowId = windows.get(0).windowId();
    System.out.println("Window ID: " + windowId);
} else {
    System.out.println("Window not found.");
}
return windowId;
}

// Create an AWS SSM document to use in this scenario.
public static void createSSMDoc(SsmClient ssmClient, String docName) {
    // Create JSON for the content
    String jsonData = ""
    {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                    "runCommand": [
                        "echo 'Hello, world!'"
                    ]
                }
            }
        ]
    }
    """;
}
```

```
try {
    CreateDocumentRequest request = CreateDocumentRequest.builder()
        .content(jsonData)
        .name(docName)
        .documentType(DocumentType.COMMAND)
        .build();

    // Create the document.
    CreateDocumentResponse response = ssmClient.createDocument(request);
    System.out.println("The status of the document is " +
response.documentDescription().status());

} catch (DocumentAlreadyExistsException e) {
    System.err.println("The document already exists. Moving on." );
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void describeOpsItems(SsmClient ssmClient, String key) {
    try {
        OpsItemFilter filter = OpsItemFilter.builder()
            .key(OpsItemFilterKey.OPS_ITEM_ID)
            .values(key)
            .operator(OpsItemFilterOperator.EQUAL)
            .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
            .maxResults(10)
            .opsItemFilters(filter)
            .build();

        DescribeOpsItemsResponse itemsResponse =
ssmClient.describeOpsItems(itemsRequest);
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
        for (OpsItemSummary item : items) {
            System.out.println("The item title is " + item.title() +" and the
status is "+item.status().toString());
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```



```
    }  
  }  
  
  public static void deleteOpsItem(SsmClient ssmClient, String opsId) {  
    try {  
      DeleteOpsItemRequest deleteOpsItemRequest =  
DeleteOpsItemRequest.builder()  
        .opsItemId(opsId)  
        .build();  
  
      ssmClient.deleteOpsItem(deleteOpsItemRequest);  
      System.out.println(opsId + " Opsitem was deleted");  
  
    } catch (SsmException e) {  
      System.err.println(e.getMessage());  
      System.exit(1);  
    }  
  }  
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [CommandInvocations](#)
 - [CreateDocument](#)
 - [CreateMaintenanceWindow](#)
 - [CreateOpsItem](#)
 - [DeleteMaintenanceWindow](#)
 - [SendCommand](#)
 - [UpdateOpsItem](#)

Exemples d'Amazon Textract utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Textract.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service

individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

AnalyzeDocument

L'exemple de code suivant montre comment utiliser `AnalyzeDocument`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <sourceDoc>\s

                Where:
                sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        analyzeDoc(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
```

```
        .bytes(sourceBytes)
        .build();

    List<FeatureType> featureTypes = new ArrayList<FeatureType>();
    featureTypes.add(FeatureType.FORMS);
    featureTypes.add(FeatureType.TABLES);

    AnalyzeDocumentRequest analyzeDocumentRequest =
    AnalyzeDocumentRequest.builder()
        .featureTypes(featureTypes)
        .document(myDoc)
        .build();

    AnalyzeDocumentResponse analyzeDocument =
    textractClient.analyzeDocument(analyzeDocumentRequest);
    List<Block> docInfo = analyzeDocument.blocks();
    Iterator<Block> blockIterator = docInfo.iterator();

    while (blockIterator.hasNext()) {
        Block block = blockIterator.next();
        System.out.println("The block type is " +
    block.blockType().toString());
    }

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [AnalyzeDocument](#) à la section Référence des AWS SDK for Java 2.x API.

DetectDocumentText

L'exemple de code suivant montre comment utiliser `DetectDocumentText`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Détecte le texte d'un document d'entrée.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <sourceDoc>\s

                Where:
                sourceDoc - The path where the document is located (must be an
                image, for example, C:/AWS/book.png).\s
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceDoc = args[0];
    Region region = Region.US_EAST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    detectDocText(textractClient, sourceDoc);
    textractClient.close();
}

public static void detectDocText(TextractClient textractClient, String
sourceDoc) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes.
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
            .document(myDoc)
            .build();

        // Invoke the Detect operation.
        DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
        List<Block> docInfo = textResponse.blocks();
        for (Block block : docInfo) {
            System.out.println("The block type is " +
block.blockType().toString());
        }

        DocumentMetadata documentMetadata = textResponse.documentMetadata();
```

```

        System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

Détecte le texte d'un document situé dans un compartiment Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <docName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket that contains the
document.\s

```

```
        docName - The document name (must be an image, i.e., book.png).
\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String docName = args[1];
    Region region = Region.US_WEST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    detectDocTextS3(textractClient, bucketName, docName);
    textractClient.close();
}

public static void detectDocTextS3(TextractClient textractClient, String
bucketName, String docName) {
    try {
        S3Object s3Object = S3Object.builder()
            .bucket(bucketName)
            .name(docName)
            .build();

        // Create a Document object and reference the s3Object instance.
        Document myDoc = Document.builder()
            .s3Object(s3Object)
            .build();

        DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
            .document(myDoc)
            .build();

        DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
        for (Block block : textResponse.blocks()) {
            System.out.println("The block type is " +
block.blockType().toString());
        }
    }
}
```



```
    }

    DocumentMetadata documentMetadata = textResponse.documentMetadata();
    System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextractException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectDocumentText](#) à la section Référence des AWS SDK for Java 2.x API.

StartDocumentAnalysis

L'exemple de code suivant montre comment utiliser `StartDocumentAnalysis`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <docName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket that contains the
document.\s
                docName - The document name (must be an image, for example,
book.png).\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
            .region(region)
            .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
        String status = getJobResults(textractClient, jobId);
        System.out.println("The job status is " + status);
        textractClient.close();
    }
}
```

```
public static String startDocAnalysisS3(TextractClient textractClient, String
bucketName, String docName) {
    try {
        List<FeatureType> myList = new ArrayList<>();
        myList.add(FeatureType.TABLES);
        myList.add(FeatureType.FORMS);

        S3Object s3Object = S3Object.builder()
            .bucket(bucketName)
            .name(docName)
            .build();

        DocumentLocation location = DocumentLocation.builder()
            .s3Object(s3Object)
            .build();

        StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
            .documentLocation(location)
            .featureTypes(myList)
            .build();

        StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

        // Get the job ID
        String jobId = response.jobId();
        return jobId;

    } catch (TextractException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

private static String getJobResults(TextractClient textractClient, String jobId)
{
    boolean finished = false;
    int index = 0;
    String status = "";

    try {
        while (!finished) {
```

```
        GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
        .jobId(jobId)
        .maxResults(1000)
        .build();

        GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
        status = response.jobStatus().toString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(index + " status is: " + status);
            Thread.sleep(1000);
        }
        index++;
    }

    return status;

} catch (InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [StartDocumentAnalysis](#) à la section Référence des AWS SDK for Java 2.x API.

Exemples d'Amazon Transcribe utilisant le SDK pour Java 2.x

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK for Java 2.x aide d'Amazon Transcribe.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés et dans des exemples interservices.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir une tâche spécifique en appelant plusieurs fonctions au sein d'un même service.

Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

ListTranscriptionJobs

L'exemple de code suivant montre comment utiliser `ListTranscriptionJobs`.

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listTranscriptionJobs(transcribeClient);
    }

    public static void listTranscriptionJobs(TranscribeClient transcribeClient)
    {
        ListTranscriptionJobsRequest listJobsRequest =
        ListTranscriptionJobsRequest.builder()
            .build();

        transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
```

```

        .flatMap(response -> response.transcriptionJobSummaries().stream())
        .forEach(jobSummary -> {
            System.out.println("Job Name: " +
jobSummary.transcriptionJobName());
            System.out.println("Job Status: " +
jobSummary.transcriptionJobStatus());
            System.out.println("Output Location: " +
jobSummary.outputLocationType());
            // Add more information as needed

            // Retrieve additional details for the job if necessary
            GetTranscriptionJobResponse jobDetails =
transcribeClient.getTranscriptionJob(
                GetTranscriptionJobRequest.builder()
                    .transcriptionJobName(jobSummary.transcriptionJobName())
                    .build());

            // Display additional details
            System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
            System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
            // Add more details as needed

            System.out.println("-----");
        });
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [ListTranscriptionJobs](#) à la section Référence des AWS SDK for Java 2.x API.

StartTranscriptionJob

L'exemple de code suivant montre comment utiliser `StartTranscriptionJob`.

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }
    }
}
```

```
        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString())
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records stream successfully ===");
            })
            .subscriber(event -> {
                List<Result> results = ((TranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {
                        System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
    }
}
```



```
        }
    })
    .build();
}

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);
```

```
SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
    this.subscriber = s;
    this.inputStream = inputStream;
}

@Override
public void request(long n) {
    if (n <= 0) {
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (Exception e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
```

```
        try {
            len = inputStream.read(audioBytes);

            if (len <= 0) {
                audioBuffer = ByteBuffer.allocate(0);
            } else {
                audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
            }
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

        return audioBuffer;
    }

    private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
        return AudioEvent.builder()
            .audioChunk(SdkBytes.fromByteBuffer(bb))
            .build();
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StartTranscriptionJob](#) à la section Référence des AWS SDK for Java 2.x API.

Scénarios

Transcription d'un fichier audio et obtention de données sur la tâche

L'exemple de code suivant illustre comment :

- Démarrez une tâche de transcription avec Amazon Transcribe.
- Attendez que la tâche se termine.
- Obtenez l'URI dans lequel la transcription est stockée.

Pour plus d'informations, consultez la section [Getting started with Amazon Transcribe](#).

SDK pour Java 2.x

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Transcrit un fichier PCM.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
    InterruptedException {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  <file> \n\n" +
            "Where:\n" +
            "  file - the location of a PCM file to transcribe. In this
example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
            .region(REGION)
            .build();
    }
}
```

```
        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromFile(file)),
    getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromFile(String file) {
    try {
        File inputFile = new File(file);
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;

    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US)
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
            System.out.println("Error Occurred: " + sw.toString());
        })
        .onComplete(() -> {
            System.out.println("=== All records stream successfully ===");
        })
        .subscriber(event -> {
```

```
        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                }
            }
        })
        .build();
    }

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
        this.subscriber = s;
    }
}
```

```
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }

    @Override
    public void cancel() {
        executor.shutdown();
    }

    private ByteBuffer getNextEvent() {
        ByteBuffer audioBuffer = null;
        byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

        int len = 0;
        try {
            len = inputStream.read(audioBytes);
        }
```

```
        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

Transcrit l'audio en streaming depuis le microphone de l'ordinateur.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }
}
```



```
}

private static InputStream getStreamFromMic() throws LineUnavailableException {

    // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
    int sampleRate = 16000;
    AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

    if (!AudioSystem.isLineSupported(info)) {
        System.out.println("Line not supported");
        System.exit(0);
    }

    TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
    line.open(format);
    line.start();

    InputStream audioStream = new AudioInputStream(line);
    return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
        });
}
```

```
        System.out.println("Error Occurred: " + sw.toString());
    })
    .onComplete(() -> {
        System.out.println("=== All records stream successfully ===");
    })
    .subscriber(event -> {
        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                }
            }
        })
    .build();
}

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
```

```
        this.currentSubscription.cancel();
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    }
    s.onSubscribe(currentSubscription);
}
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
    {
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }
}
```

```
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
 - [GetTranscriptionJob](#)
 - [StartTranscriptionJob](#)

Exemples multiservices utilisant le SDK pour Java 2.x

Les exemples d'applications suivants utilisent le AWS SDK for Java 2.x pour fonctionner sur plusieurs applications Services AWS.

Les exemples multiservices ciblent un niveau d'expérience avancé pour vous aider à commencer à créer des applications.

Exemples

- [Créer une application pour soumettre des données à une table DynamoDB](#)
- [Créer un chatbot Amazon Lex pour engager les visiteurs de votre site Web](#)
- [Créer une application de publication et d'abonnement qui traduit les messages](#)
- [Créer une application Web qui envoie et récupère des messages à l'aide d'Amazon SQS](#)
- [Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes](#)
- [Créer une application web pour suivre les données DynamoDB](#)
- [Créer un outil de suivi des éléments Amazon Redshift.](#)
- [Créer un outil de suivi des éléments de travail sans serveur Aurora](#)
- [Créer une application qui analyse les commentaires des clients et synthétise le son](#)
- [Déterminez le PPE dans les images avec Amazon Rekognition à l'aide d'un SDK AWS](#)
- [Déterminez des objets dans des images avec Amazon Rekognition à l'aide d'un SDK AWS](#)
- [Déterminez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un SDK AWS](#)
- [Surveillez les performances d'Amazon DynamoDB à l'aide d'un SDK AWS](#)
- [Publiez des messages Amazon SNS dans les files d'attente Amazon SQS à l'aide d'un SDK AWS](#)
- [Utiliser API Gateway pour appeler une fonction Lambda](#)
- [Utiliser les fonctions Step Functions pour invoquer des fonctions Lambda](#)
- [Utilisent des événements planifiés pour appeler une fonction Lambda](#)

Créer une application pour soumettre des données à une table DynamoDB

SDK pour Java 2.x

Indique comment créer une application Web dynamique qui envoie des données à l'aide de l'API Java Amazon DynamoDB et envoie un message texte à l'aide de l'API Java Amazon Simple Notification Service.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- DynamoDB
- Amazon SNS

Créez un chatbot Amazon Lex pour engager les visiteurs de votre site Web

SDK pour Java 2.x

Montre comment utiliser l'API Amazon Lex pour créer un Chatbot au sein d'une application Web afin d'engager les visiteurs de votre site Web.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Créer une application de publication et d'abonnement qui traduit les messages

SDK pour Java 2.x

Indique comment utiliser l'API Java Amazon Simple Notification Service pour créer une application Web dotée de fonctionnalités d'abonnement et de publication. De plus, cet exemple d'application traduit également des messages.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour obtenir le code source complet et les instructions sur la façon de configurer et d'exécuter l'exemple utilisant l'API Java Async, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon Translate

Créez une application Web qui envoie et récupère des messages à l'aide d'Amazon SQS

SDK pour Java 2.x

Montre comment utiliser l'API Amazon SQS pour développer une API Spring REST qui envoie et récupère des messages.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Comprehend
- Amazon SQS

Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes

SDK pour Java 2.x

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur [AWS Community](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Créer une application web pour suivre les données DynamoDB

SDK pour Java 2.x

Montre comment utiliser l'API Amazon DynamoDB pour créer une application web dynamique qui suit les données de travail DynamoDB.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- DynamoDB
- Amazon SES

Créer un outil de suivi des éléments Amazon Redshift.

SDK pour Java 2.x

Montre comment créer une application web qui suit et génère des rapports sur les éléments de travail stockés dans une base de données Amazon Redshift.

Pour obtenir le code source complet et les instructions sur la façon de configurer une API Spring REST qui interroge les données Amazon Redshift et pour une utilisation par une application React, consultez l'exemple complet sur [GitHub](#)

Les services utilisés dans cet exemple

- Amazon Redshift
- Amazon SES

Créer un outil de suivi des éléments de travail sans serveur Aurora

SDK pour Java 2.x

Montre comment créer une application web qui suit et génère des rapports sur les éléments de travail stockés dans une base de données Amazon RDS.

Pour obtenir le code source complet et les instructions sur la façon de configurer une API Spring REST qui interroge les données Amazon Aurora Serverless et pour une utilisation par une application React, consultez l'exemple complet sur [GitHub](#).

Pour obtenir le code source complet et les instructions sur la façon de configurer et d'exécuter un exemple utilisant l'API JDBC, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Créez une application qui analyse les commentaires des clients et synthétise le son

SDK pour Java 2.x

Cet exemple d'application analyse et stocke les cartes de commentaires des clients. Plus précisément, elle répond aux besoins d'un hôtel fictif situé à New York. L'hôtel reçoit les commentaires des clients dans différentes langues sous la forme de cartes de commentaires physiques. Ces commentaires sont chargés dans l'application via un client Web. Après avoir chargé l'image d'une carte de commentaires, les étapes suivantes se déroulent :

- Le texte est extrait de l'image à l'aide d'Amazon Textract.
- Amazon Comprehend détermine le sentiment du texte extrait et sa langue.
- Le texte extrait est traduit en anglais à l'aide d'Amazon Translate.
- Amazon Polly synthétise un fichier audio à partir du texte extrait.

L'application complète peut être déployée avec AWS CDK. Pour le code source et les instructions de déploiement, consultez le projet dans [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Détectez le PPE dans les images avec Amazon Rekognition à l'aide d'un SDK AWS

SDK pour Java 2.x

Montre comment créer une AWS Lambda fonction qui détecte les images à l'aide d'un équipement de protection individuelle.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Détectez des objets dans des images avec Amazon Rekognition à l'aide d'un SDK AWS

SDK pour Java 2.x

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui, avec Amazon Rekognition, permet d'identifier des objets par catégorie dans des images stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un SDK AWS

SDK pour Java 2.x

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui détecte les visages et les objets dans des vidéos stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Surveillez les performances d'Amazon DynamoDB à l'aide d'un SDK AWS

SDK pour Java 2.x

Cet exemple montre comment configurer une application Java pour surveiller les performances de DynamoDB. L'application envoie des données métriques vers CloudWatch lesquelles vous pouvez surveiller les performances.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- CloudWatch
- DynamoDB

Publiez des messages Amazon SNS dans les files d'attente Amazon SQS à l'aide d'un SDK AWS

SDK pour Java 2.x

Présente la messagerie avec des rubriques et des files d'attente avec Amazon Simple Notification Service (Amazon SNS) et Amazon Simple Queue Service (Amazon SQS).

Pour obtenir le code source complet et les instructions illustrant la messagerie avec des rubriques et des files d'attente dans Amazon SNS et Amazon SQS, consultez l'exemple complet sur [GitHub](#)

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon SQS

Utiliser API Gateway pour appeler une fonction Lambda

SDK pour Java 2.x

Montre comment créer une AWS Lambda fonction à l'aide de l'API d'exécution Lambda Java. Cet exemple fait appel à différents AWS services pour réaliser un cas d'utilisation spécifique. Cet exemple montre comment créer une fonction Lambda invoquée par Amazon API Gateway qui analyse une table Amazon DynamoDB à la recherche d'anniversaires professionnels et utilise Amazon Simple Notification Service (Amazon SNS) pour envoyer un message texte à vos employés qui les félicitent à leur date d'anniversaire.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB

- Lambda
- Amazon SNS

Utiliser les fonctions Step Functions pour invoquer des fonctions Lambda

SDK pour Java 2.x

Montre comment créer un flux de travail AWS sans serveur en utilisant AWS Step Functions et le AWS SDK for Java 2.x. Chaque étape du flux de travail est implémentée à l'aide d'une AWS Lambda fonction.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Utilisent des événements planifiés pour appeler une fonction Lambda

SDK pour Java 2.x

Montre comment créer un événement EventBridge planifié Amazon qui invoque une AWS Lambda fonction. Configurez EventBridge pour utiliser une expression cron afin de planifier le moment où la fonction Lambda est invoquée. Dans cet exemple, vous créez une fonction Lambda à l'aide de l'API d'exécution Lambda. Cet exemple fait appel à différents AWS services pour réaliser un cas d'utilisation spécifique. Cet exemple montre comment créer une application qui envoie un message texte mobile à vos employés pour les féliciter à leur date d'anniversaire.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- DynamoDB

- EventBridge
- Lambda
- Amazon SNS

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 2.x](#)
- [Utilisation du protocole TLS dans le SDK pour Java](#)
- [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](#)

Protection des données dans la version AWS SDK for Java 2.x

Le [modèle de responsabilité AWS partagée](#) de s'applique à la protection des données dans AWS SDK for Java. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure

mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des 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 [Questions fréquentes \(FAQ\) sur la confidentialité des données](#). Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée [AWS et RGPD \(Règlement général sur la protection des données\)](#) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou 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 Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans 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 fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Name (Nom). Cela inclut lorsque vous travaillez avec le SDK for Java ou un Services AWS autre outil à l'aide de la console, de l'API AWS CLI AWS ou des SDK. Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Utilisation du protocole TLS dans le SDK pour Java

AWS SDK for Java utilise les fonctionnalités TLS de sa plate-forme Java sous-jacente. Dans cette rubrique, nous présentons des exemples utilisant l'implémentation OpenJDK utilisée par [Amazon Corretto 17](#).

Pour fonctionner avec Services AWS, le JDK sous-jacent doit prendre en charge une version minimale de TLS 1.2, mais le protocole TLS 1.3 est recommandé.

Les utilisateurs doivent consulter la documentation de la plate-forme Java qu'ils utilisent avec le SDK pour savoir quelles versions de TLS sont activées par défaut et comment activer et désactiver des versions TLS spécifiques.

Comment vérifier les informations de version de TLS

[À l'aide d'OpenJDK, le code suivant montre l'utilisation de SSLContext pour imprimer les versions TLS/SSL prises en charge.](#)

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

Par exemple, Amazon Corretto 17 (OpenJDK) produit le résultat suivant.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

Pour voir la liaison SSL en action et quelle version de TLS est utilisée, vous pouvez utiliser la propriété système `javax.net.debug`.

Par exemple, exécutez une application Java qui utilise le protocole TLS.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

L'application enregistre la poignée de main SSL de la manière suivante.

```
...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.221 EST|ClientHello.java:641|Produced
  ClientHello handshake message (
  "ClientHello": {
    "client version"      : "TLSv1.2",
  }
  ...
```

```
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.295 EST|ServerHello.java:888|Consuming
ServerHello handshake message (
"ServerHello": {
  "server version"      : "TLSv1.2",
  ...
```

Appliquer une version minimale de TLS

Le SDK pour Java préfère toujours la dernière version TLS prise en charge par la plateforme et le service. Si vous souhaitez appliquer une version minimale spécifique de TLS, consultez la documentation de votre plateforme Java.

Pour les machines virtuelles Java basées sur OpenJDK, vous pouvez utiliser la propriété système `jdk.tls.client.protocols`

Par exemple, si vous souhaitez que les clients du service SDK de votre application utilisent le protocole TLS 1.2, même si le protocole TLS 1.3 est disponible, indiquez la propriété système suivante.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

AWS Mise à niveau des points de terminaison d'API vers TLS 1.2

Consultez ce billet de [blog](#) pour plus d'informations sur le passage des points de terminaison d' AWS API à TLS 1.2 pour la version minimale.

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'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 votre Compte AWS qui possède des autorisations spécifiques pour une seule personne ou 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 Comptes AWS 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 déterminer 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 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 fournissent des étapes 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 plusieurs 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](#).

Migrez de la version 1.x vers la version 2.x du AWS SDK for Java

Le AWS SDK for Java 2.x est une réécriture majeure de la base de code 1.x construite sur Java 8+. Cette version comprend de nombreuses mises à jour, dont une amélioration de la cohérence et de la facilité d'utilisation et une mise en œuvre robuste de l'immuabilité. Cette section décrit les principales fonctionnalités nouvelles de la version 2.x et fournit des conseils sur la façon de migrer votre code vers la version 2.x à partir de la version 1.x.

Rubriques

- [Nouveautés de la version 2](#)
- [step-by-step Instructions de migration avec exemple](#)
- [Nom du package vers les mappages Artifactid Maven](#)
- [Quelle est la différence entre la version AWS SDK for Java 1.x et la version 2.x](#)
- [Utilisation du kit SDK for Java 1.x side-by-side](#)

Nouveautés de la version 2

- Vous pouvez configurer vos propres clients HTTP. Consultez la section [Configuration du transport HTTP](#).
- Les clients asynchrones offrent un support d'E/S non bloquant et renvoyant des objets. `CompletableFuture` Voir [Programmation asynchrone](#).
- Les opérations qui renvoient plusieurs pages ont des réponses paginées automatiquement. De cette façon, vous pouvez concentrer votre code sur ce qu'il faut faire avec la réponse, sans avoir à vérifier et à obtenir les pages suivantes. Voir [Pagination](#).
- Les performances des AWS Lambda fonctions à l'heure de démarrage du SDK sont améliorées. Consultez la section [Améliorations des performances du SDK à l'heure de démarrage](#).
- La version 2.x prend en charge une nouvelle méthode raccourcie pour créer des demandes.

Exemple

```
dynamoDbClient.putItem(request -> request.tableName(TABLE))
```


Pour plus de détails sur les nouvelles fonctionnalités et pour voir des exemples de code spécifiques, consultez les autres sections de ce guide.

- [Quick Start](#)
- [Configuration](#)
- [Exemples de code pour le AWS SDK for Java 2.x](#)
- [Utiliser le SDK](#)
- [Sécurité pour AWS SDK for Java](#)

step-by-step Instructions de migration avec exemple

Cette section fournit un step-by-step guide pour migrer votre application qui utilise actuellement le SDK pour Java v1.x vers le SDK pour Java 2.x. La première partie présente un aperçu des étapes suivies par un exemple détaillé de migration.

Les étapes décrites ici décrivent une migration d'un cas d'utilisation normal, dans lequel l'application appelle à l' Services AWS aide de clients de service pilotés par un modèle. Si vous devez migrer du code qui utilise des API de niveau supérieur, telles que [S3 Transfer Manager](#) ou [CloudFrontPresigning](#), reportez-vous à la section sous la [the section called “Quelles sont les différences entre 1.x et 2.x”](#) table des matières.

L'approche décrite ici est une suggestion. Vous pouvez utiliser d'autres techniques et tirer parti des fonctionnalités d'édition de code de votre IDE pour obtenir le même résultat.

Présentation des étapes

1. Commencez par ajouter le SDK pour Java 2.x BOM

En ajoutant l'élément Maven BOM (Bill of Materials) pour le SDK pour Java 2.x à votre fichier POM, vous vous assurez que toutes les dépendances v2 dont vous avez besoin proviennent de la même version. Votre POM peut contenir à la fois des dépendances v1 et v2. Cela vous permet de migrer votre code progressivement plutôt que de le modifier en une seule fois.

SDK pour Java 2.x BOM

```
<dependencyManagement>
  <dependencies>
```

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>bom</artifactId>
  <version>2.24.3</version>
  <type>pom</type>
  <scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

Vous pouvez trouver la [dernière version](#) sur le référentiel central de Maven.

2. Rechercher dans les fichiers les instructions d'importation de classe v1

En scannant les fichiers de votre application à la recherche des Service_ID utilisés dans les importations de la version 1, vous trouverez les Service_ID uniques utilisés. Un SERVICE_ID est un nom court et unique pour un Service AWS cognito identity C'est le cas par exemple du SERVICE_ID pour Amazon Cognito Identity.

3. Déterminez les dépendances Maven de la version 2 à partir des instructions d'importation de la version 1

Une fois que vous avez trouvé tous les Service_ID uniques de la v1, vous pouvez déterminer l'artefact Maven correspondant à la dépendance v2 en vous référant à [the section called "Nom du package vers les mappages ArtifactID"](#)

4. Ajouter des éléments de dépendance v2 au fichier POM

Mettez à jour le fichier Maven POM avec les éléments de dépendance déterminés à l'étape 3.

5. Dans les fichiers Java, passez progressivement des classes v1 aux classes v2

Lorsque vous remplacez les classes v1 par des classes v2, apportez les modifications nécessaires pour prendre en charge l'API v2, par exemple en utilisant des générateurs plutôt que des constructeurs et en utilisant des getters et des setters fluides.

6. Supprimer les dépendances Maven v1 du POM et les importations v1 des fichiers

Après avoir migré votre code pour utiliser les classes v2, supprimez toutes les importations v1 restantes des fichiers et toutes les dépendances de votre fichier de compilation.

7. Refactorisez le code pour utiliser les améliorations de l'API v2

Une fois le code compilé et passé avec succès les tests, vous pouvez tirer parti des améliorations apportées à la version 2, telles que l'utilisation d'un autre client HTTP ou de différents paginateurs pour simplifier le code. Il s'agit d'une étape facultative.

Exemple de migration

Dans cet exemple, nous migrons une application qui utilise le SDK pour Java v1 et accède Services AWS à plusieurs d'entre eux. Nous travaillons en détail sur la méthode v1 suivante à l'étape 5. Il s'agit d'une méthode dans une classe qui contient huit méthodes et l'application compte 32 classes.

méthode v1 pour migrer

Seules les importations du SDK v1 sont répertoriées ci-dessous à partir du fichier Java.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
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.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple requests.
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.
```

```

        for (final Reservation r : result.getReservations()) {
            for (final Instance instance : r.getInstances()) {
                LOGGER.info("Examining instanceId: " + instance.getInstanceId());
                // if instance is in a running state, add it to runningInstances
list.
                if (RUNNING_STATES.contains(instance.getState().getName())) {
                    runningInstances.add(instance);
                }
            }
        }
    } while (result.getNextToken() != null);
} catch (final AmazonEC2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}

```

1. Ajouter la version v2 de Maven BOM

Ajoutez la nomenclature Maven pour le SDK pour Java 2.x au POM avec toutes les autres dépendances de la section. `dependencyManagement` Si votre fichier POM contient la nomenclature de la version 1 du SDK, laissez-la pour le moment. Il sera supprimé ultérieurement.

Gestion des dépendances POM dès le départ

```

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.example</groupId>                <!--Existing dependency in POM. -->
      <artifactId>bom</artifactId>
      <version>1.3.4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId> <!--Existing v1 BOM dependency. -->

```

```
<version>1.11.1000</version>
<type>pom</type>
<scope>import</scope>
</dependency>
...
<dependency>
  <groupId>software.amazon.awssdk</groupId> <!--Add v2 BOM dependency. -->
  <artifactId>bom</artifactId>
  <version>2.24.3</version>
  <type>pom</type>
  <scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

2. Rechercher dans les fichiers les instructions d'importation de classe v1

Recherchez dans le code de l'application les occurrences uniques de `import com.amazonaws.services`. Cela nous aide à déterminer les dépendances v1 utilisées par le projet. Si votre application possède un fichier Maven POM dont les dépendances v1 sont répertoriées, vous pouvez utiliser ces informations à la place.

Dans cet exemple, nous utilisons la commande [ripgrep\(rg\)](#) pour effectuer une recherche dans la base de code.

À la racine de votre base de code, exécutez la `ripgrep` commande suivante. Une fois `ripgrep` les instructions d'importation trouvées, elles sont redirigées vers les `uniq` commandes `cut` et `sort`, et pour isoler les `Service_ID`.

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

Pour cette application, les `Service_ID` suivants sont enregistrés sur la console.

```
autoscaling
cloudformation
ec2
identitymanagement
```

Cela indique qu'au moins une occurrence de chacun des noms de package suivants a été utilisée dans les `import` instructions. Quatre de nos objectifs, les noms de classe individuels n'ont pas d'importance. Il suffit de trouver les `Service_ID` utilisés.

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

3. Déterminez les dépendances Maven de la version 2 à partir des instructions d'importation de la version 1

Les Service_ID de la v1 que nous avons isolés à l'étape 2, par exemple `autoscaling` et, `cloudformation` peuvent être mappés sur le même SERVICE_ID de la version v2 pour la plupart. Étant donné que l'artifactid Maven v2 correspond au SERVICE_ID dans la plupart des cas, vous disposez des informations dont vous avez besoin pour ajouter des blocs de dépendance à votre fichier POM.

Le tableau suivant montre comment déterminer les dépendances de la v2.

v1 SERVICE_ID correspond à...	v2 SERVICE_ID correspond à...	Dépendance Maven v2
nom du package	nom du package	
ec2 com.amazonaws.services. ec2 .*	ec2 software.amazon.awssdk.services. ec2 .*	<pre><dependency> <groupId>software.amazon.awssdk</groupId> <artifactId> ec2</artifactId> </dependency></pre>
mise à l'échelle automatique com.amazonaws.services. autoscaling .*	mise à l'échelle automatique software.amazon.awssdk.services. autoscaling .*	<pre><dependency> <groupId>software.amazon.awssdk</groupId> <artifactId> autoscaling </artifactId> </dependency></pre>
cloudformation	cloudformation	<pre><dependency></pre>

v1 SERVICE_ID correspond à...	v2 SERVICE_ID correspond à...	Dépendance Maven v2
nom du package com.amazonaws.services. cloudformation .*	nom du package software.amazon.awssdk. cloudformation .*	<pre><groupId>software.amazon.awssdk</groupId> <artifactId> cloudformation </artifactId> </dependency></pre>
gestion de l'identité* com.amazonaws.services. identitymanagement .*	je suis* software.amazon.awssdk. iam .*	<pre><dependency> <groupId>software.amazon.awssdk</groupId> <artifactId> iam</artifactId> </dependency></pre>

* Le iam mappage identitymanagement vers est une exception lorsque le SERVICE_ID diffère d'une version à l'autre. Reportez-vous [the section called “Nom du package vers les mappages ArtifactID”](#) aux exceptions si Maven ou Gradle ne peuvent pas résoudre la dépendance à la v2.

4. Ajouter des éléments de dépendance v2 au fichier POM

À l'étape 3, nous avons déterminé les quatre blocs de dépendance à ajouter au fichier POM. Nous n'avons pas besoin d'ajouter de version car nous avons spécifié la nomenclature à l'étape 1. Une fois les importations ajoutées, notre fichier POM contient les éléments de dépendance suivants.

```
...
<dependencies>
  ...
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>autoscaling</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam</artifactId>
```

```
</dependency>
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>cloudformation</artifactId>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>ec2</artifactId>
</dependency>
...
</dependencies>
...
```

5. Dans les fichiers Java, passez progressivement des classes v1 aux classes v2

Dans la méthode de migration, nous voyons

- Un client du service EC2 de `com.amazonaws.services.ec2.AmazonEC2Client`
- Plusieurs classes de modèles EC2 ont été utilisées. Par exemple `DescribeInstancesRequest` et `DescribeInstancesResult`.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
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.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds)
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
```



```
do {
    // DescribeInstancesResponse is a paginated response, so use tokens with
multiple re
    result = ec2.describeInstances(request);
    request.setNextToken(result.getNextToken()); // Prepare request for next
page.
    for (final Reservation r : result.getReservations()) {
        for (final Instance instance : r.getInstances()) {
            LOGGER.info("Examining instanceId: " + instance.getInstanceId());
            // if instance is in a running state, add it to runningInstances
list.
                if (RUNNING_STATES.contains(instance.getState().getName())) {
                    runningInstances.add(instance);
                }
            }
        }
    } while (result.getNextToken() != null);
} catch (final AmazonEC2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}
...
```

Notre objectif est de remplacer toutes les importations v1 par des importations v2. Nous procédons un cours à la fois.

a. Remplacer la déclaration d'importation ou le nom de classe

Nous voyons que le premier paramètre de la `describeRunningInstances` méthode est une `AmazonEC2Client` instance v1. Effectuez l'une des actions suivantes :

- Remplacez le fichier d'importation `com.amazonaws.services.ec2.AmazonEC2Client` par `software.amazon.awssdk.services.ec2.Ec2Client` et `AmazonEC2Client` remplacez-le par `Ec2Client`.
- Changez le type de paramètre en `Ec2Client` et laissez l'IDE nous demander l'importation correcte. Notre IDE nous demandera d'importer la classe v2 car les noms des clients diffèrent...

`AmazonEC2Client` et `Ec2Client`. Cette approche ne fonctionne pas si le nom de classe est le même dans les deux versions.

b. Remplacer les classes de modèles v1 par des équivalents v2

Après le passage à la `v2Ec2Client`, si nous utilisons un IDE, nous voyons des erreurs de compilation dans l'instruction suivante.

```
result = ec2.describeInstances(request);
```

L'erreur de compilation résulte de l'utilisation d'une instance de v1 `DescribeInstancesRequest` comme paramètre de la `Ec2Client` `describeInstances` méthode v2. Pour résoudre le problème, effectuez les instructions de remplacement ou d'importation suivantes.

replace	avec
<pre>import com.amazonaws.services.ec2.model.DescribeInstancesRequest</pre>	<pre>import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest</pre>

c. Changez les constructeurs v1 en constructeurs v2.

Nous voyons toujours des erreurs de compilation car il [n'y a pas de constructeur sur les classes v2](#). Pour corriger cela, apportez les modifications suivantes.

modification	to
<pre>final DescribeInstancesRequest request = new DescribeInstancesRequest().withInstanceIds(instanceIdsCopy);</pre>	<pre>final DescribeInstancesRequest request = DescribeInstancesRequest.builder().instanceIds(instanceIdsCopy).build();</pre>

d. Remplacer les objets de ***Result** réponse v1 par des ***Response** équivalents v2

Une différence constante entre la v1 et la v2 est que tous les [objets de réponse de la v2 se terminent par *Response au lieu de *Result](#). Remplacez l'`DescribeInstancesResult` importation v1 par l'importation v2, `DescribeInstancesResponse`.

d. Apporter des modifications à l'API

La déclaration suivante nécessite quelques modifications.

```
request.setNextToken(result.getNextToken());
```

Dans la version 2, [les méthodes setter](#) n'utilisent pas le `set` ou avec `prefix`. Les méthodes Getter préfixées par `get` ont également disparu dans le SDK pour Java 2.x

Les classes de modèles, telles que l'`requestinstance`, sont immuables dans la version 2, nous devons donc en créer une nouvelle `DescribeInstancesRequest` avec un générateur.

Dans la version 2, l'instruction devient la suivante.

```
request = DescribeInstancesRequest.builder()
    .nextToken(result.nextToken())
    .build();
```

d. Répétez jusqu'à ce que la méthode soit compilée avec les classes v2

Continuez avec le reste du code. Remplacez les importations v1 par des importations v2 et corrigez les erreurs de compilation. Reportez-vous à la [référence de l'API v2](#) et à la [référence What's different](#) si nécessaire.

Après avoir migré cette méthode unique, nous avons le code v2 suivant.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
```

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
...
private static List<Instance> getRunningInstances(Ec2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(instanceIds)
            .build();
        DescribeInstancesResponse result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens
with multiple re
            result = ec2.describeInstances(request);
            request = DescribeInstancesRequest.builder() // Prepare request for
next page.
                .nextToken(result.nextToken())
                .build();
            for (final Reservation r : result.reservations()) {
                for (final Instance instance : r.instances()) {
                    // if instance is in a running state, add it to
runningInstances list.
                    if (RUNNING_STATES.contains(instance.state().nameAsString())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.nextToken() != null);
    } catch (final Ec2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
        if (exception.awsErrorDetails().errorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
            throw exception;
        }
    }
    return runningInstances;
}
```

...

Comme nous migrons une seule méthode dans un fichier Java avec huit méthodes, nous avons un mélange d'importations v1 et v2 au fur et à mesure que nous travaillons sur le fichier. Nous avons ajouté les six dernières instructions d'importation au fur et à mesure de l'exécution des étapes.

Une fois que nous aurons migré tout le code, il n'y aura plus d'instructions d'importation v1.

6. Supprimer les dépendances Maven v1 du POM et les importations v1 des fichiers

Après avoir migré tout le code v1 du fichier, nous avons les instructions d'importation du SDK v2 suivantes.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.regions.ServiceMetadata;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.InstanceStateName;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
```

Après avoir migré tous les fichiers de notre application, nous n'avons plus besoin des dépendances v1 dans notre fichier POM. Supprimez la nomenclature v1 de la dependencyManagement section, si vous en utilisez, ainsi que tous les blocs de dépendance v1.

7. Refactorisez le code pour utiliser les améliorations de l'API v2

Pour l'extrait que nous avons migré, nous pouvons éventuellement utiliser un paginateur v2 et laisser le SDK gérer les demandes de données supplémentaires basées sur des jetons.

Nous pouvons remplacer l'intégralité de la do clause par la suivante.

```
DescribeInstancesIterable responses =
    ec2.describeInstancesPaginator(request);

    responses.reservations().stream()
```

```

        .forEach(reservation -> reservation.instances()
            .forEach(instance -> {
                if
(RUNNING_STATES.contains(instance.state().nameAsString())) {
                    runningInstances.put(instance.instanceId(),
instance);
                }
            }));

```

Nom du package vers les mappages ArtifactID Maven

Lorsque vous migrez votre projet Maven ou Gradle de la version v1 du SDK pour Java vers la version v2, vous devez déterminer les dépendances à ajouter à votre fichier de compilation. L'approche décrite à [l'<u>the section called “tep-by-step Instructions S”</u>](#) étape 3 utilise les noms des packages dans les instructions d'importation comme point de départ pour déterminer les dépendances (sous forme d'artifactIds) à ajouter à votre fichier de compilation.

Vous pouvez utiliser les informations de cette rubrique pour associer les noms des packages v1 aux ArtifactID de la version v2.

Convention de dénomination commune utilisée dans les noms de packages et les ArtifactID de Maven

Le tableau suivant indique la convention de dénomination commune que les SDK utilisent pour un SERVICE_ID donné. Un SERVICE_ID est un identifiant unique pour un. Service AWS Par exemple, le SERVICE_ID du service Amazon S3 est s3 et cognitoidentity est le SERVICE_ID d'Amazon Cognito Identity.

nom du package v1 (déclaration d'importa tion)	Artifactid v1	Artifactid v2	nom du package v2 (instruction d'importa tion)
com.amazonaws.serv ices.Service_ID	aws-java-sdk-IDENT IFIANT DE SERVICE	IDENTIFIANT DE SERVICE	Software.Amazon.AW SSDK.Services.Serv ice_ID

Exemple pour Amazon Cognito Identity (SERVICE_ID :) ***cognitoidentity***

nom du package v1 (déclaration d'importa tion)	Artifac ti d v1	Artifac ti d v2	nom du package v2 (instruction d'importa tion)
com.amazonaws.serv ices. identité cognito	aws-java-sdk- identité cognitive	identité cognito	software.amazon.aw ssdk.services. identité cognito

Différences entre SERVICE_ID

Dans la v1

Dans certains cas, le SERVICE_ID est différent entre le nom du package et l'Artifactid du même service. Par exemple, la ligne CloudWatch Metrics du tableau suivant indique qu'il `metrics` s'agit du SERVICE_ID dans le nom du package mais `cloudwatchmetrics` du SERVICE_ID de l'artifactid.

Dans la v2

Il n'y a aucune différence entre le SERVICE_ID utilisé dans les noms de package et les Artifactid.

Entre v1 et v2

Pour la majorité des services, le SERVICE_ID de la version v2 est identique au SERVICE_ID de la version 1 dans les noms de package et dans les artifactids. Le `cognitoentity` SERVICE_ID, comme indiqué dans le tableau précédent, en est un exemple. Cependant, certains Service_ID diffèrent entre les SDK, comme indiqué dans le tableau suivant.

Un SERVICE_ID en gras dans l'une des colonnes v1 indique qu'il est différent du SERVICE_ID utilisé dans la v2.

Nom du service	nom du package v1	Artifac ti d v1	Artifac ti d v2	nom du package v2
	Tous les noms de packages commencent par <code>com.amazo naws.serv</code>	Tous les Artifac ti D sont inclus dans des balises, comme	Tous les Artifac ti D sont inclus dans des balises, comme	Tous les noms de packages commencent par <code>software. amazon.aw</code>

Nom du service	nom du package v1	Artifactid v1	Artifactid v2	nom du package v2
	ices le nom indiqué dans la première ligne.	indiqué dans la première ligne.	indiqué dans la première ligne.	ssdk le nom indiqué dans la première ligne.
API Gateway	com.amazonaws.services.apigateway	<artifactId>aws-java-sdk-passerelle API</artifactId>	<artifactId>passerelle apigateway</artifactId>	software.amazon.awssdk.services.apigateway
Registre des applications	apregistry	apregistry	registre des applications du catalogue de services	registre des applications du catalogue de services
Application Discovery	découverte d'applications	découverte	découverte d'applications	découverte d'applications
Augmented AI Runtime	durée d'antenne accrue	durée d'antenne accrue	exécution de sagemaker2i	exécution de sagemaker2i
Certificate Manager	gestionnaire de certificats	acm	acm	acm
CloudControl API	API de contrôle du cloud	API de contrôle du cloud	contrôle du cloud	contrôle du cloud
CloudSearch	CloudSearchv2	cloudsearch	cloudsearch	cloudsearch
CloudSearch Domaine	domaine de recherche dans le cloud	recherche dans le cloud	domaine de recherche dans le cloud	domaine de recherche dans le cloud
CloudWatch Évènements	événements CloudWatch	événements	événements CloudWatch	événements CloudWatch

Nom du service	nom du package v1	Artifactid v1	Artifactid v2	nom du package v2
CloudWatch De toute évidence	Cloudwatch, de toute évidence	Cloudwatch, de toute évidence	evidently	evidently
CloudWatch Journaux	journaux	journaux	journaux de surveillance du cloud	journaux de surveillance du cloud
CloudWatch Métriques	métriques	métriques de CloudWatch	cloudwatch	cloudwatch
CloudWatch Rhum	rhum Cloudwatch	rhum Cloudwatch	rum	rum
Fournisseur d'identité Cognito	cognitoidp	cognitoidp	fournisseur d'identité cognito	fournisseur d'identité cognito
Campagne Connect	campagne connect	campagne connect	connectez les campagnes	connectez les campagnes
Connect Wisdom	connecter la sagesse	connecter la sagesse	wisdom	wisdom
Database Migration Service	service de migration de base de données	dms	migration de base de données	migration de base de données
DataZone	zone de données	zone de données externe	zone de données	zone de données
DynamoDB	dynamodbv2	dynamodb	dynamodb	dynamodb
Système de fichiers Elastic	système de fichiers élastique	efs	efs	efs
Elastic Map Reduce	elasticmapreduce	emr	emr	emr

Nom du service	nom du package v1	Artifactid v1	Artifactid v2	nom du package v2
Glue DataBrew	infusion de colle	infusion de colle	databrew	databrew
Rôles Anywhere IAM	je joue des rôles partout	je joue des rôles partout	rolesanywhere	rolesanywhere
Gestion des identités	gestion de l'identité	iam	iam	iam
Données relatives à l'IoT	données IoT	iot	plan de données IoT	plan de données IoT
Kinesis Analytics	kinesisanalytics	kinesis	kinesisanalytics	kinesisanalytics
Kinesis Firehose	tuyau à incendie Kinesis	kinesis	firehose	firehose
Canaux de signalisation vidéo Kinesis	canaux de signalisation vidéo kinesis	canaux de signalisation vidéo kinesis	signalisation vidéo kinesis	signalisation vidéo kinesis
Lex	lexruntime	lex	lexruntime	lexruntime
À l'affût de la vision	attention à la vision	attention à la vision	lookoutvision	lookoutvision
Modernisation du mainframe	modernisation de l'ordinateur central	modernisation de l'ordinateur central	m2	m2
Marketplace Metering	mesure du marché	service de mesure du marché	mesure du marché	mesure du marché
Grafana géré	grafana géré	grafana géré	grafana	grafana
Mechanical Turk	mturk	Requêteur mécanique	mturk	mturk

Nom du service	nom du package v1	Artifactid v1	Artifactid v2	nom du package v2
Migration Hub Strategy Recommendations	recommandations relatives à la stratégie du pôle migratoire	recommandations relatives à la stratégie du pôle migratoire	stratégie du pôle migratoire	stratégie du pôle migratoire
Studio agile	studio agile	studio agile	nimble	nimble
5G privée	5 g privés	5 g privés	réseaux privés	réseaux privés
Prometheus	prométhée	prométhée	ampli	ampli
Corbeille	corbeille	corbeille	rbin	rbin
API de données Redshift	API de données Redshift	API de données Redshift	données redshift	données redshift
Route 53	Route 53 domaines	route53	Route 53 domaines	Route 53 domaines
Gestionnaire Sage Maker Edge	gestionnaire de sagemakeredge	gestionnaire de sagemakeredge	sagemakeredge	sagemakeredge
Jeton de sécurité	jeton de sécurité	sts	sts	sts
Migration de serveurs	migration de serveurs	migration de serveurs	sms	sms
Courrier électronique simple	e-mail simple	ses	ses	ses
Courrier électronique simple V2	e-mail simple v2	sesv2	sesv2	sesv2
Gestion simplifiée des systèmes	gestion simple des systèmes	ssm	ssm	ssm

Nom du service	nom du package v1	Artifactid v1	Artifactid v2	nom du package v2
Flux de travail simplifié	flux de travail simple	flux de travail simple	swf	swf
Step Functions	fonctions d'étape	fonctions d'étape	sfn	sfn

Quelle est la différence entre la version AWS SDK for Java 1.x et la version 2.x

Cette section décrit les principaux changements à prendre en compte lors de la conversion d'une application de la AWS SDK for Java version 1.x à la version 2.x.

Modification du nom du package

Un changement notable entre le SDK pour Java 1.x et le SDK pour Java 2.x est le changement de nom du package. Les noms des packages commencent par `software.amazon.awssdk` dans le SDK 2.x, alors que le SDK 1.x utilise `com.amazonaws`

Ces mêmes noms différencient les artefacts Maven du SDK 1.x au SDK 2.x. Les artefacts Maven du SDK 2.x utilisent le GroupID, tandis que le SDK 1.x utilise le `software.amazon.awssdk` GroupID. `com.amazonaws`

Il arrive parfois que votre code nécessite une `com.amazonaws` dépendance pour un projet qui, par ailleurs, n'utilise que des artefacts du SDK 2.x. Par exemple, lorsque vous travaillez avec le côté serveur AWS Lambda. Cela a été expliqué dans la section [Configurer un projet Apache Maven](#) plus haut dans ce guide.

Note

Le SDK 1.x contient plusieurs noms de packages. v2 v2 Dans ce cas, l'utilisation de signifie généralement que le code du package est conçu pour fonctionner avec la version 2 du service.

Comme le nom complet du package commence par `com.amazonaws`, il s'agit de composants du SDK 1.x. Voici des exemples de ces noms de packages dans le SDK 1.x :

- `com.amazonaws.services.dynamodbv2`

- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

Ajouter la version 2.x à votre projet

Maven est la méthode recommandée pour gérer les dépendances lors de l'utilisation de la version AWS SDK for Java 2.x. Pour ajouter des composants de la version 2.x à votre projet, mettez à jour votre `pom.xml` fichier avec une dépendance au SDK.

Exemple

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```

Vous pouvez également [utiliser les versions 1.x et 2.x lorsque side-by-side](#) vous migrez votre projet vers la version 2.x.

POJO immuables

Les clients et les objets de demande et de réponse d'opération sont désormais immuables et ne peuvent pas être modifiés après création. Pour réutiliser une variable de demande ou de réponse, vous devez créer un objet pour l'attribuer.

Exemple de mise à jour d'un objet de demande dans la version 1.x

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

Exemple de mise à jour d'un objet de demande dans la version 2.x

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
    .nextToken(response.nextToken())
    .build();
```

Méthodes Setter et Getter

Dans la version AWS SDK for Java 2.x, les noms des méthodes setter n'incluent pas le préfixe `set` ou `with`. Par exemple, `*.withEndpoint()` c'est maintenant `*.endpoint()`.

Les noms des méthodes Getter n'utilisent pas le `get` préfixe.

Exemple de l'utilisation des méthodes setter dans la version 1.x

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion("us-east-1")
    .build();
```

Exemple de l'utilisation des méthodes setter dans la version 2.x

```
DynamoDbClient client = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .build();
```

Exemple de l'utilisation des méthodes getter dans la version 1.x

```
String token = request.getNextToken();
```

Exemple de l'utilisation des méthodes getter dans la version 2.x

```
String token = request.nextToken();
```

Noms des classes de modèles

Les noms des classes de modèles qui représentent les réponses des services se terminent par `Response` la version 2 au lieu de `Result` celle utilisée par la version 1.

Exemple de noms de classes qui représentent une réponse dans la version 1

```
CreateApiKeyResult
AllocateAddressResult
```

Exemple de noms de classes qui représentent une réponse dans la version 2

```
CreateApiKeyResponse
AllocateAddressResponse
```

État de migration des bibliothèques et des utilitaires

SDK pour bibliothèques et utilitaires Java

Le tableau suivant répertorie l'état de migration des bibliothèques et des utilitaires pour le SDK for Java.

Nom de la version 1.12.x	Nom de la version 2.x	Depuis la version 2.x
DynamoDBMapper	DynamoDbEnhancedClient	2.12.0
Programmes d'attente	Programmes d'attente	2.15.0
CloudFrontUrlSigner, CloudFrontCookieSigner	CloudFrontUtilities	2,18,33
TransferManager	S3 TransferManager	2.19.0
Client de métadonnées EC2	Client de métadonnées EC2	2,19,29

Nom de la version 1.12.x	Nom de la version 2.x	Depuis la version 2.x
Analyseur d'URI S3	Analyseur d'URI S3	2,20,41
Générateur de politiques IAM	Générateur de politiques IAM	2,20,126
Mise en mémoire tampon côté client Amazon SQS	Traitement automatique des demandes par lots	pas encore publié
Écouteurs de progression	Écouteurs de progression	pas encore publié

Bibliothèques associées

Le tableau suivant répertorie les bibliothèques publiées séparément mais compatibles avec le SDK for Java 2.x.

Nom utilisé avec la version 2.x du SDK pour Java	Depuis la version
Client de chiffrement Amazon S3	3,0.0.1
AWS Client de chiffrement de base de données pour DynamoDB	3,0.0.2

¹ Le client de chiffrement pour Amazon S3 est disponible en utilisant la dépendance Maven suivante.

```
<dependency>
  <groupId>software.amazon.encryption.s3</groupId>
  <artifactId>amazon-s3-encryption-client-java</artifactId>
  <version>3.x</version>
</dependency>
```

² Le client AWS de chiffrement de base de données pour DynamoDB est disponible en utilisant la dépendance Maven suivante.

```
<dependency>
  <groupId>software.amazon.cryptography</groupId>
  <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
```



```
<version>3.x</version>
</dependency>
```

Détails de migration pour les bibliothèques et les utilitaires

- [Gestionnaire de transfert S3](#)
- [Utilitaire de métadonnées EC2](#)
- [CloudFrontprésignant](#)
- [Analyse d'URI S3](#)

Modifications du client

Créateurs de clients

Vous devez créer tous les clients à l'aide de la méthode du générateur client. Les constructeurs ne sont plus disponibles.

Exemple de la création d'un client dans la version 1.x

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

Exemple de la création d'un client dans la version 2.x

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

Noms des classes de clients

Tous les noms de classes de clients sont désormais entièrement encadrés par un chameau et ne sont plus préfixés par `Amazon`. Ces modifications correspondent aux noms utilisés dans l'AWS CLI.

Exemple des noms de classe dans la version 1.x

```
AmazonDynamoDB
AWSACMPAAsyncClient
```

Exemple des noms de classe dans la version 2.x

```
DynamoDbClient
AcmAsyncClient
```

Changements de nom de classe du client

Client 1.x	Client 2.x
<code>com.amazonaws.services.acmpca.AWSACMPCAAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.acmpca.AWSACMPCAClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessAsyncClient</code>	<code>software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessAsyncClient</code>
<code>com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessClient</code>	<code>software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingAsyncClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamAsyncClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamAsyncClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncAsyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncAsyncClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaAsyncClient</code>	<code>software.amazon.awssdk.services.athena.AthenaAsyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaClient</code>	<code>software.amazon.awssdk.services.athena.AthenaClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient</code>
<code>com.amazonaws.services.batch.AWSBatchAsyncClient</code>	<code>software.amazon.awssdk.services.batch.BatchAsyncClient</code>
<code>com.amazonaws.services.batch.AWSBatchClient</code>	<code>software.amazon.awssdk.services.batch.BatchClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsAsyncClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsAsyncClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9AsyncClient</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9AsyncClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9Client</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9Client</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.codestar.AWSCodeStarAsyncClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarAsyncClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient</code>
<code>com.amazonaws.services.cognitoidentityprovider.AWSIdentityProviderAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient</code>
<code>com.amazonaws.services.cognitoidentityprovider.AWSIdentityProviderClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendAsyncClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.comprehend.AmazonComprehendClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendClient</code>
<code>com.amazonaws.services.config.AmazonConfigAsyncClient</code>	<code>software.amazon.awssdk.services.config.ConfigAsyncClient</code>
<code>com.amazonaws.services.config.AmazonConfigClient</code>	<code>software.amazon.awssdk.services.config.ConfigClient</code>
<code>com.amazonaws.services.connect.AmazonConnectAsyncClient</code>	<code>software.amazon.awssdk.services.connect.ConnectAsyncClient</code>
<code>com.amazonaws.services.connect.AmazonConnectClient</code>	<code>software.amazon.awssdk.services.connect.ConnectClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineAsyncClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.dax.AmazonDaxAsyncClient</code>	<code>software.amazon.awssdk.services.dax.DaxAsyncClient</code>
<code>com.amazonaws.services.dax.AmazonDaxClient</code>	<code>software.amazon.awssdk.services.dax.DaxClient</code>
<code>com.amazonaws.services.devicefarm.AWSDeviceFarmAsyncClient</code>	<code>software.amazon.awssdk.services.devicefarm.DeviceFarmAsyncClient</code>
<code>com.amazonaws.services.devicefarm.AWSDeviceFarmClient</code>	<code>software.amazon.awssdk.services.devicefarm.DeviceFarmClient</code>
<code>com.amazonaws.services.directconnect.AmazonDirectConnectAsyncClient</code>	<code>software.amazon.awssdk.services.directconnect.DirectConnectAsyncClient</code>
<code>com.amazonaws.services.directconnect.AmazonDirectConnectClient</code>	<code>software.amazon.awssdk.services.directconnect.DirectConnectClient</code>
<code>com.amazonaws.services.directory.AWSDirectoryServiceAsyncClient</code>	<code>software.amazon.awssdk.services.directory.DirectoryAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.directory.AWSDirectoryServiceClient</code>	<code>software.amazon.awssdk.services.directory.DirectoryClient</code>
<code>com.amazonaws.services.dlm.AmazonDLMAsyncClient</code>	<code>software.amazon.awssdk.services.dlm.DlmAsyncClient</code>
<code>com.amazonaws.services.dlm.AmazonDLMClient</code>	<code>software.amazon.awssdk.services.dlm.DlmClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2AsyncClient</code>	<code>software.amazon.awssdk.services.ec2.Ec2AsyncClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2Client</code>	<code>software.amazon.awssdk.services.ec2.Ec2Client</code>
<code>com.amazonaws.services.ecr.AmazonECRAsyncClient</code>	<code>software.amazon.awssdk.services.ecr.EcrAsyncClient</code>
<code>com.amazonaws.services.ecr.AmazonECRClient</code>	<code>software.amazon.awssdk.services.ecr.EcrClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.ecs. AmazonECSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ecs.EcsAsyncClient</code>
<code>com.amazonaws.services.ecs. AmazonECSClient</code>	<code>software.amazon.awssdk.serv ices.ecs.EcsClient</code>
<code>com.amazonaws.services.eks. AmazonEKSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksAsyncClient</code>
<code>com.amazonaws.services.eks. AmazonEKSClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheAs yncClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eAsyncClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkAsyncClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkAsyncClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemAsyncClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsAsyncClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient</code>	<code>software.amazon.awssdk.services.emr.EmrAsyncClient</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient</code>	<code>software.amazon.awssdk.services.emr.EmrClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient</code>
<code>com.amazonaws.services.fms.AWSFMSAsyncClient</code>	<code>software.amazon.awssdk.services.fms.FmsAsyncClient</code>
<code>com.amazonaws.services.fms.AWSFMSClient</code>	<code>software.amazon.awssdk.services.fms.FmsClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftAsyncClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierAsyncClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierAsyncClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierClient</code>
<code>com.amazonaws.services.glue.AWSGlueAsyncClient</code>	<code>software.amazon.awssdk.services.glue.GlueAsyncClient</code>
<code>com.amazonaws.services.glue.AWSGlueClient</code>	<code>software.amazon.awssdk.services.glue.GlueClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassAsyncClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassAsyncClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyClient</code>
<code>com.amazonaws.services.health.AWSHealthAsyncClient</code>	<code>software.amazon.awssdk.services.health.HealthAsyncClient</code>
<code>com.amazonaws.services.health.AWSHealthClient</code>	<code>software.amazon.awssdk.services.health.HealthClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient</code>	<code>software.amazon.awssdk.services.iam.IamAsyncClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient</code>	<code>software.amazon.awssdk.services.iam.IamClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportAsyncClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportAsyncClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorAsyncClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorAsyncClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.iot.AWSIoTAsyncClient</code>	<code>software.amazon.awssdk.services.iot.IotAsyncClient</code>
<code>com.amazonaws.services.iot.AWSIoTClient</code>	<code>software.amazon.awssdk.services.iot.IotClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataAsyncClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataAsyncClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisAsyncClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisAsyncClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseAsyncClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient</code>	Non pris en charge
<code>com.amazonaws.services.kms.AWSKMSAsyncClient</code>	<code>software.amazon.awssdk.services.kms.KmsAsyncClient</code>
<code>com.amazonaws.services.kms.AWSKMSClient</code>	<code>software.amazon.awssdk.services.kms.KmsClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaAsyncClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaAsyncClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailAsyncClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailAsyncClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailClient</code>
<code>com.amazonaws.services.logs.AWSLogsAsyncClient</code>	<code>software.amazon.awssdk.services.logs.LogsAsyncClient</code>
<code>com.amazonaws.services.logs.AWSLogsClient</code>	<code>software.amazon.awssdk.services.logs.LogsClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.macie.AsyncClient</code>	<code>software.amazon.awssdk.services.macie.MacieAsyncClient</code>
<code>com.amazonaws.services.macie.MacieClient</code>	<code>software.amazon.awssdk.services.macie.MacieClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertClient</code>
<code>com.amazonaws.services.mediaalive.AWSMediaLiveAsyncClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveAsyncClient</code>
<code>com.amazonaws.services.mediaalive.AWSMediaLiveClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.mediataylor.AWSMediaTailorAsyncClient</code>	<code>software.amazon.awssdk.services.mediataylor.MediaTailorAsyncClient</code>
<code>com.amazonaws.services.mediataylor.AWSMediaTailorClient</code>	<code>software.amazon.awssdk.services.mediataylor.MediaTailorClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubClient</code>
<code>com.amazonaws.services.mobile.AWSMobileAsyncClient</code>	<code>software.amazon.awssdk.services.mobile.MobileAsyncClient</code>
<code>com.amazonaws.services.mobile.AWSMobileClient</code>	<code>software.amazon.awssdk.services.mobile.MobileClient</code>
<code>com.amazonaws.services.mq.AmazonMQAsyncClient</code>	<code>software.amazon.awssdk.services.mq.MqAsyncClient</code>
<code>com.amazonaws.services.mq.AmazonMQClient</code>	<code>software.amazon.awssdk.services.mq.MqClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkAsyncClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkAsyncClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneAsyncClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.neptune.AmazonNeptuneClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmAsyncClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsAsyncClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsAsyncClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsClient</code>
<code>com.amazonaws.services.pi.AWSPIAsyncClient</code>	<code>software.amazon.awssdk.services.pi.PiAsyncClient</code>
<code>com.amazonaws.services.pi.AWSPIClient</code>	<code>software.amazon.awssdk.services.pi.PiClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.pinpoint.AmazonPinpointClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointClient</code>
<code>com.amazonaws.services.polly.AmazonPollyAsyncClient</code>	<code>software.amazon.awssdk.services.polly.PollyAsyncClient</code>
<code>com.amazonaws.services.polly.AmazonPollyClient</code>	<code>software.amazon.awssdk.services.polly.PollyClient</code>
<code>com.amazonaws.services.pricing.AWS PricingAsyncClient</code>	<code>software.amazon.awssdk.services.pricing.PricingAsyncClient</code>
<code>com.amazonaws.services.pricing.AWS PricingClient</code>	<code>software.amazon.awssdk.services.pricing.PricingClient</code>
<code>com.amazonaws.services.rds.AmazonRDSAsyncClient</code>	<code>software.amazon.awssdk.services.rds.RdsAsyncClient</code>
<code>com.amazonaws.services.rds.AmazonRDSClient</code>	<code>software.amazon.awssdk.services.rds.RdsClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftAsyncClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftAsyncClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionAsyncClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53AsyncClient</code>	<code>software.amazon.awssdk.services.route53.Route53AsyncClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53Client</code>	<code>software.amazon.awssdk.services.route53.Route53Client</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsClient</code>
<code>com.amazonaws.services.s3.AmazonS3Client</code>	<code>software.amazon.awssdk.services.s3.S3Client</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient</code>	<code>software.amazon.awssdk.services.sts.StsAsyncClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient</code>	<code>software.amazon.awssdk.services.sts.StsClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient</code>	<code>software.amazon.awssdk.services.sms.SmsAsyncClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationClient</code>	<code>software.amazon.awssdk.services.sms.SmsClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient</code>
<code>com.amazonaws.services.shield.AWSShieldAsyncClient</code>	<code>software.amazon.awssdk.services.shield.ShieldAsyncClient</code>
<code>com.amazonaws.services.shield.AWSShieldClient</code>	<code>software.amazon.awssdk.services.shield.ShieldClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.simpledb.AmazonSimpleDBAsyncClient</code>	<code>software.amazon.awssdk.services.simpledb.SimpleDbAsyncClient</code>
<code>com.amazonaws.services.simpledb.AmazonSimpleDBClient</code>	<code>software.amazon.awssdk.services.simpledb.SimpleDbClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient</code>	<code>software.amazon.awssdk.services.ses.SesAsyncClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient</code>	<code>software.amazon.awssdk.services.ses.SesClient</code>
<code>com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementAsyncClient</code>	<code>software.amazon.awssdk.services.ssm.SsmAsyncClient</code>
<code>com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementClient</code>	<code>software.amazon.awssdk.services.ssm.SsmClient</code>
<code>com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowAsyncClient</code>	<code>software.amazon.awssdk.services.swf.SwfAsyncClient</code>
<code>com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClient</code>	<code>software.amazon.awssdk.services.swf.SwfClient</code>
<code>com.amazonaws.services.snowball.AmazonSnowballAsyncClient</code>	<code>software.amazon.awssdk.services.snowball.SnowballAsyncClient</code>
<code>com.amazonaws.services.snowball.AmazonSnowballClient</code>	<code>software.amazon.awssdk.services.snowball.SnowballClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.sns. AmazonSNSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sns.SnsAsyncClient</code>
<code>com.amazonaws.services.sns. AmazonSNSClient</code>	<code>software.amazon.awssdk.serv ices.sns.SnsClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsAsyncClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsA syncClient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnAsyncClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsC lient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yAsyncClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayAsyncClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportAsyncClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportAsyncClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportClient</code>
<code>com.amazonaws.services.tran scribe.AmazonTranscribeAsyn cClient</code>	<code>software.amazon.awssdk.serv ices.transcribe.TranscribeA syncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.transcribe.AmazonTranscribeClient</code>	<code>software.amazon.awssdk.services.transcribe.TranscribeClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateAsyncClient</code>	<code>software.amazon.awssdk.services.translate.TranslateAsyncClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateClient</code>	<code>software.amazon.awssdk.services.translate.TranslateClient</code>
<code>com.amazonaws.services.waf.AWSWAFAsyncClient</code>	<code>software.amazon.awssdk.services.waf.WafAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFClient</code>	<code>software.amazon.awssdk.services.waf.WafClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalAsyncClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailAsyncClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailAsyncClient</code>

Client 1.x	Client 2.x
<code>com.amazonaws.services.workmail.AmazonWorkMailClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkspacesAsyncClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkspacesClient</code>
<code>com.amazonaws.services.xray.AWSXRayAsyncClient</code>	<code>software.amazon.awssdk.services.xray.XRayAsyncClient</code>
<code>com.amazonaws.services.xray.AWSXRayClient</code>	<code>software.amazon.awssdk.services.xray.XRayClient</code>

Valeurs par défaut de création du client

Dans la version 2.x, les modifications suivantes ont été apportées à la logique de création du client par défaut.

- La chaîne de fournisseurs d'informations d'identification par défaut pour S3 n'inclut plus d'informations d'identification anonymes. Vous devez spécifier manuellement un accès anonyme à S3 à l'aide du `AnonymousCredentialsProvider`.
- Les variables d'environnement suivantes liées à la création du client par défaut sont différentes.

1.x	2.x
<code>AWS_CBOR_DISABLED</code>	<code>CBOR_ENABLED</code>
<code>AWS_ION_BINARY_DISABLE</code>	<code>BINARY_ION_ENABLED</code>

- Les propriétés système suivantes relatives à la création de clients par défaut sont différentes.

1.x	2.x
<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>
<code>com.amazonaws.sdk.disableCbor</code>	<code>aws.cborEnabled</code>
<code>com.amazonaws.sdk.disableIoBinary</code>	<code>aws.binaryIonEnabled</code>

- La version 2.x ne prend pas en charge les propriétés système suivantes.

1.x

`com.amazonaws.sdk.disableCertChecking`

`com.amazonaws.sdk.enableDefaultMetrics`

`com.amazonaws.sdk.enableThrottledRetry`

`com.amazonaws.regions.RegionUtils.fileOverride`

`com.amazonaws.regions.RegionUtils.disableRemote`

`com.amazonaws.services.s3.disableImplicitGlobalClients`

`com.amazonaws.sdk.enableInRegionOptimizedMode`

- Le chargement de la configuration des régions à partir d'un `endpoints.json` fichier personnalisé n'est plus pris en charge.

Configuration du client

Dans la version 1.x, la configuration du client SDK était modifiée en définissant une `ClientConfiguration` instance sur le client ou le générateur de clients. Dans la version 2.x, la configuration du client est scindée en classes de configuration distinctes. Avec les classes de

configuration distinctes, vous pouvez configurer différents clients HTTP pour les clients asynchrones et les clients synchrones, tout en utilisant la même `ClientOverrideConfiguration` classe.

Exemple de la configuration du client dans la version 1.x

```
AmazonDynamoDBClientBuilder.standard()
    .withClientConfiguration(clientConfiguration)
    .build()
```

Exemple de la configuration du client synchrone dans la version 2.x

```
ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder();

ApacheHttpClient.Builder httpClientBuilder =
    ApacheHttpClient.builder()
        .proxyConfiguration(proxyConfig.build());

ClientOverrideConfiguration.Builder overrideConfig =
    ClientOverrideConfiguration.builder();

DynamoDbClient client =
    DynamoDbClient.builder()
        .httpClientBuilder(httpClientBuilder)
        .overrideConfiguration(overrideConfig.build())
        .build();
```

Exemple de la configuration du client asynchrone dans la version 2.x

```
NettyNioAsyncHttpClient.Builder httpClientBuilder =
    NettyNioAsyncHttpClient.builder();

ClientOverrideConfiguration.Builder overrideConfig =
    ClientOverrideConfiguration.builder();

ClientAsyncConfiguration.Builder asyncConfig =
    ClientAsyncConfiguration.builder();

DynamoDbAsyncClient client =
    DynamoDbAsyncClient.builder()
        .httpClientBuilder(httpClientBuilder)
        .overrideConfiguration(overrideConfig.build())
        .asyncConfiguration(asyncConfig.build())
```

```
.build();
```

Clients HTTP

Changements notables

- Dans la version 2.x, vous pouvez modifier le client HTTP à utiliser lors de l'exécution en spécifiant une implémentation à l'aide `clientBuilder.httpClientBuilder` de.
- Lorsque vous transmettez un client HTTP `clientBuilder.httpClient` à l'aide d'un générateur de clients de service, le client HTTP n'est pas fermé par défaut s'il ferme le client de service. Cela vous permet de partager des clients HTTP entre des clients de service.
- Les clients HTTP asynchrones utilisent désormais des E/S non bloquantes.
- Certaines opérations utilisent désormais HTTP/2 pour améliorer les performances.

Modifications des paramètres

Paramètre	1.x	Synchronisation 2.x, Apache	Async 2.x, Netty
	<pre>ClientCon figuration clientConfig = new ClientCon figuration()</pre>	<pre>ApacheHtt pClient.B uilder httpClien tBuilder = ApacheHtt pClient.b uilder()</pre>	<pre>NettyNioA syncHttpC lient.Builder httpClient tBuilder = NettyNioA syncHttpC lient.builder()</pre>
Nombre maximum de connexions	<pre>clientCon fig.setMa xConnecti ons(...) clientCon fig.withM axConnect ions(...)</pre>	<pre>httpClien tBuilder. maxConnec tions(...)</pre>	<pre>httpClien tBuilder. maxConcur rency(...)</pre>

Paramètre	1.x	Synchronisation 2.x, Apache	Async 2.x, Netty
Délai de connexion	<pre>clientConfig.setConnectionTimeout(...) clientConfig.withConnectionTimeout(...)</pre>	<pre>httpClientBuilder.connectionTimeout(...) httpClientBuilder.connectionAcquisitionTimeout(...)</pre>	<pre>httpClientBuilder.connectionTimeout(...)</pre>
Expiration du socket	<pre>clientConfig.setSocketTimeout(...) clientConfig.withSocketTimeout(...)</pre>	<pre>httpClientBuilder.socketTimeout(...)</pre>	<pre>httpClientBuilder.writeTimeout(...) httpClientBuilder.readTimeout(...)</pre>
Connexion TTL	<pre>clientConfig.setConnectionTTL(...) clientConfig.withConnectionTTL(...)</pre>	<pre>httpClientBuilder.connectionTimeToLive(...)</pre>	<pre>httpClientBuilder.connectionTimeToLive(...)</pre>

Paramètre	1.x	Synchronisation 2.x, Apache	Async 2.x, Netty
Connexion maximale en veille	<pre>clientConfig.setConnectionMaxIdleMillis(...) clientConfig.withConnectionMaxIdleMillis(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>
Valider après inactivité	<pre>clientConfig.setValidateAfterInactivityMillis(...) clientConfig.withValidateAfterInactivityMillis(...)</pre>	Non pris en charge (fonctionnalité de demande)	Non pris en charge (fonctionnalité de demande)
Adresse locale	<pre>clientConfig.setLocalAddress(...) clientConfig.withLocalAddress(...)</pre>	<pre>httpClientBuilder.localAddress(...)</pre>	Non pris en charge
Expect-continue activé	<pre>clientConfig.setUseExpectContinue(...) clientConfig.withUseExpectContinue(...)</pre>	<pre>httpClientBuilder.expectContinueEnabled(...)</pre>	Non pris en charge (fonctionnalité de demande)

Paramètre	1.x	Synchronisation 2.x, Apache	Async 2.x, Netty
Reaper Connection	<pre>clientConfig.setUseReaper(...) clientConfig.withReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDBClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>	<pre>DynamoDBAsyncClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

Proxys clients HTTP

Paramètres	1.x	Synchronisation 2.x, Apache	Async 2.x, Netty
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>
Hôte proxy	<pre>clientConfig.setProxyHost(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.host(...)</pre>

Paramètres	1.x	Synchronisation 2.x, Apache	Async 2.x, Netty
	<code>clientConfig.withProxyHost(...)</code>		
Port proxy	<code>clientConfig.setProxyPort(...)</code> <code>clientConfig.withProxyPort(...)</code>	<code>proxyConfig.endpoint(...)</code> Le port proxy est intégré dans endpoint	<code>proxyConfig.port(...)</code>
Nom d'utilisateur du proxy	<code>clientConfig.setProxyUsername(...)</code> <code>clientConfig.withProxyUsername(...)</code>	<code>proxyConfig.username(...)</code>	<code>proxyConfig.username(...)</code>
Mot de passe proxy	<code>clientConfig.setProxyPassword(...)</code> <code>clientConfig.withProxyPassword(...)</code>	<code>proxyConfig.password(...)</code>	<code>proxyConfig.password(...)</code>
Domaine proxy	<code>clientConfig.setProxyDomain(...)</code> <code>clientConfig.withProxyDomain(...)</code>	<code>proxyConfig.ntlmDomain(...)</code>	Non pris en charge (fonctionnalité de demande)

Paramètres	1.x	Synchronisation 2.x, Apache	Async 2.x, Netty
Station de travail proxy	<pre>clientConfig.setProxyWorkspace(...) clientConfig.withProxyWorkstation(...)</pre>	<pre>proxyConfig.ntlmWorkstation(...)</pre>	Non pris en charge (fonctionnalité de demande)
Méthodes d'authentification par proxy	<pre>clientConfig.setProxyAuthenticationMethods(...) clientConfig.withProxyAuthenticationMethods(...)</pre>	Non pris en charge	Non pris en charge (fonctionnalité de demande)
Authentification proxy de base préemptive	<pre>clientConfig.setPreemptiveBasicProxyAuth(...) clientConfig.withPreemptiveBasicProxyAuth(...)</pre>	<pre>proxyConfig.preemptiveBasicAuthenticationEnabled(...)</pre>	Non pris en charge (fonctionnalité de demande)

Paramètres	1.x	Synchronisation 2.x, Apache	Async 2.x, Netty
Hôtes non proxy	<pre>clientConfig.setNonProxyHosts(...) clientConfig.withNonProxyHosts(...)</pre>	<pre>proxyConfiguration.nonProxyHosts(...)</pre>	<pre>proxyConfiguration.nonProxyHosts(...)</pre>
Désactiver le socket proxy	<pre>clientConfig.setDisableSocketProxy(...) clientConfig.withDisableSocketProxy(...)</pre>	Non pris en charge (fonctionnalité de demande)	Non pris en charge (fonctionnalité de demande)
	<pre>AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>

Déroptions du client

Paramètre	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientCon figuration()</pre>	<pre>ClientOverrideConf figuration.Builder overrideConfig = ClientOverrideConf figuration.builder()</pre>
Préfixe de l'agent utilisateur	<pre>clientConfig.setUs erAgentPrefix(...) clientConfig.with UserAgentPrefix(...)</pre>	<pre>overrideConfig.adv ancedOption(SdkAdvancedClientO ption.USER_AGENT_P REFIX, ...)</pre>
Suffixe d'agent utilisateur	<pre>clientConfig.setUs erAgentSuffix(...) clientConfig.with UserAgentSuffix(...)</pre>	<pre>overrideConfig.adv ancedOption(SdkAdvancedClientO ption.USER_AGENT_S UFFIX, ...)</pre>
Signer	<pre>clientConfig.setSi gnerOverride(...) clientConfig.withS ignerOverride(...)</pre>	<pre>overrideConfig.adv ancedOption(SdkAdvancedClientO ption.SIGNER, ...)</pre>
En-têtes supplémentaires	<pre>clientConfig.addHe ader(...) clientConfig.with Header(...)</pre>	<pre>overrideConfig.put Header(...)</pre>
Expiration de la demande	<pre>clientConfig.setRe questTimeout(...) clientConfig.withR equestTimeout(...)</pre>	<pre>overrideConfig.api CallAttemptTimeout (...)</pre>

Paramètre	1.x	2.x
Délai d'exécution du client	<pre>clientConfig.setClientExecutionTimeout(...) clientConfig.withClientExecutionTimeout(...)</pre>	<pre>overrideConfig.apiCallTimeout(...)</pre>
Utiliser Gzip	<pre>clientConfig.setUseGzip(...) clientConfig.withGzip(...)</pre>	Non pris en charge (fonctionnalité de demande)
Indication de la taille de la mémoire tampon	<pre>clientConfig.setSocketBufferSizeHints(...) clientConfig.withSocketBufferSizeHints(...)</pre>	Non pris en charge (fonctionnalité de demande)
Métadonnées de réponse au cache	<pre>clientConfig.setCacheResponseMetadata(...) clientConfig.withCacheResponseMetadata(...)</pre>	Non pris en charge (fonctionnalité de demande)
Taille du cache des métadonnées de réponse	<pre>clientConfig.setResponseMetadataCacheSize(...) clientConfig.withResponseMetadataCacheSize(...)</pre>	Non pris en charge (fonctionnalité de demande)

Paramètre	1.x	2.x
résolveur DNS	<pre>clientConfig.setDnsResolver(...) clientConfig.withDnsResolver(...)</pre>	Non pris en charge (fonctionnalité de demande)
TCP Keepalive	<pre>clientConfig.setUseTcpKeepAlive(...) clientConfig.withTcpKeepAlive(...)</pre>	<p>Cette option est désormais dans la configuration du client HTTP</p> <ul style="list-style-type: none"> - <code>ApacheHttpClient.builder().tcpKeepAlive(true)</code> - <code>NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)</code>
Aléatoire sécurisé	<pre>clientConfig.setSecureRandom(...) clientConfig.withSecureRandom(...)</pre>	Non pris en charge (fonctionnalité de demande)
	<pre>AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

Le client annule de nouvelles tentatives

Paramètre	1.x	2.x
	<pre>ClientConfiguration clientConfig =</pre>	<pre>RetryPolicy.Builder retryPolicy =</pre>

Paramètre	1.x	2.x
	<pre>new ClientConfiguration()</pre>	<pre>RetryPolicy.builder()</pre>
Nombre maximum de tentatives d'erreur	<pre>clientConfig.setMaxErrorRetry(...) clientConfig.withMaxErrorRetry(...)</pre>	<pre>retryPolicy.numRetries(...)</pre>
Utiliser des tentatives limitées	<pre>clientConfig.setUseThrottleRetries(...) clientConfig.withUseThrottleRetries(...)</pre>	Non pris en charge
Nombre maximum de tentatives consécutives avant la limitation	<pre>clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...) clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)</pre>	Non pris en charge
	<pre>AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

Clients asynchrones

Paramètre	1.x	2.x
		<pre>ClientAsyncConfiguration.Builder asyncConfig = ClientAsyncConfiguration.builder()</pre>
Exécuteur	<pre>AmazonDynamoDBAsyncClientBuilder.standard() .withExecutorFactory(...) .build()</pre>	<pre>asyncConfig.advancedOption(SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)</pre>
		<pre>DynamoDbAsyncClient.builder() .asyncConfiguration(asyncConfig) .build()</pre>

Autres modifications apportées aux clients

L'`ClientConfigurationOption` suivante de la version 1.x a été modifiée dans la version 2.x du SDK et n'a pas d'équivalent direct.

Paramètre	1.x	équivalent 2.x
Protocole	<pre>clientConfig.setProtocol(Protocol.HTTP) clientConfig.withProtocol(Protocol.HTTP)</pre>	<p>Le paramètre du protocole est HTTPS par défaut. Pour modifier le paramètre, spécifiez le protocole définissant un point de terminaison HTTP dans le générateur de clients :</p>

Paramètre	1.x	équivalent 2.x
		<pre>clientBuilder.endpointOverride(URI.create("http://..."))</pre>

Modifications apportées au fournisseur d'identifiants

Cette section fournit un mappage des changements de nom des classes et méthodes du fournisseur d'informations d'identification entre les versions 1.x et 2.x du AWS SDK for Java

Des différences notables

- Le fournisseur d'informations d'identification par défaut charge les propriétés système avant les variables d'environnement dans la version 2.x. Pour plus d'informations, consultez la section [Utilisation des informations d'identification](#).
- La méthode du constructeur est remplacé par les méthodes `create` ou `builder`.

Exemple

```
DefaultCredentialsProvider.create();
```

- L'actualisation asynchrone n'est plus définie par défaut. Vous devez la spécifier avec le `builder` du fournisseur d'informations d'identification.

Exemple

```
ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
    .asyncCredentialUpdateEnabled(true)
    .build();
```

- Vous pouvez spécifier un chemin d'accès à un fichier de profil personnalisé à l'aide du `ProfileCredentialsProvider.builder()`.

Exemple

```
ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()
```

```
.profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
    .build();
```

- Le format de fichier de profil a changé pour mieux correspondre à l'AWS CLI. Pour plus de détails, consultez [la section Configuration du AWS CLI](#) dans le guide de AWS Command Line Interface l'utilisateur.

Modifications du fournisseur d'informations d'identification mappées entre les versions 1.x et 2.x

AWSCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	com.amazonaws.auth.AWSCredentialsProvider	software.amazon.awssdk.auth.credentials.AwsCredentialsProvider
Nom de la méthode	getCredentials	resolveCredentials
Méthode non prise en charge	refresh	Non pris en charge

DefaultAWSCredentialsProviderChain

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	com.amazonaws.auth.DefaultAWSCredentialsProviderChain	software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider
Création	new DefaultAWSCredentialsProviderChain	DefaultCredentialsProvider.create

Changer de catégorie	1.x	2.x
Méthode non prise en charge	<code>getInstance</code>	Non pris en charge
Ordre de priorité des paramètres externes	Variables d'environnement avant les propriétés du système	Propriétés du système avant les variables d'environnement

AWSStaticCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.AWSStaticCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.StaticCredentialsProvider</code>
Création	<code>new AWSStaticCredentialsProvider</code>	<code>StaticCredentialsProvider.create</code>

EnvironmentVariableCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.EnvironmentVariableCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider</code>
Création	<code>new EnvironmentVariableCredentialsProvider</code>	<code>EnvironmentVariableCredentialsProvider.create</code>
Nom de la variable d'environnement	<code>AWS_ACCESS_KEY</code>	<code>AWS_ACCESS_KEY_ID</code>

Changer de catégorie	1.x	2.x
	<code>AWS_SECRET_KEY</code>	<code>AWS_SECRET_ACCESS_KEY</code>

SystemPropertiesCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.SystemPropertiesCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider</code>
Création	<code>new SystemPropertiesCredentialsProvider</code>	<code>SystemPropertiesCredentialsProvider.create</code>
Nom de la propriété du système	<code>aws.secretKey</code>	<code>aws.secretAccessKey</code>

ProfileCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.profile.ProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider</code>
Création	<code>new ProfileCredentialsProvider</code>	<code>ProfileCredentialsProvider.create</code>
Emplacement du profil personnalisé	<ul style="list-style-type: none"> variable d'environnement <code>AWS_CREDENTIAL_PROFILES_FILE</code> 	<ul style="list-style-type: none"> variable d'environnement <code>AWS_SHARED_CREDENTIALS_FILE</code>

Changer de catégorie	1.x	2.x
	<ul style="list-style-type: none"> • <code>new ProfileCredentialsProvider</code> 	<ul style="list-style-type: none"> • <code>ProfileCredentialsProvider.builder</code>

ContainerCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.ContainerCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code>
Création	<code>new ContainerCredentialsProvider</code>	<code>ContainerCredentialsProvider.create</code>
Spécifier l'actualisation asynchrone	Non pris en charge	Comportement par défaut

InstanceProfileCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.InstanceProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
Création	<code>new InstanceProfileCredentialsProvider</code>	<code>InstanceProfileCredentialsProvider.create</code>
Spécifier l'actualisation asynchrone	<code>new InstanceProfileCredentialsProvider(true)</code>	<code>InstanceProfileCredentialProvider.builder().asyncCred</code>

Changer de catégorie	1.x	2.x
		<code>essentialUpdateEnabled(true).build()</code>
Nom de la propriété du système	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>

STSAssumeRoleSessionCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider</code>
Création	<ul style="list-style-type: none"> <code>new STSAssumeRoleSessionCredentialsProvider</code> <code>new STSAssumeRoleSessionCredentialsProvider.Builder</code> 	<code>StsAssumeRoleCredentialsProvider.builder</code>
Actualisation asynchrone	Comportement par défaut	Comportement par défaut
Configuration	<code>new STSAssumeRoleSessionCredentialsProvider.Builder</code>	Configurer une <code>AssumeRoleRequest</code> demande <code>StsClient</code> et

STSSessionCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.STSSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider</code>
Création	<code>new STSAssumeRoleSessionCredentialsProvider</code>	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Actualisation asynchrone	Comportement par défaut	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Configuration	Paramètres du constructeur	Configurer une <code>GetSessionTokenRequest</code> demande <code>StsClient</code> et dans un générateur

WebIdentityFederationSessionCredentialsProvider

Changer de catégorie	1.x	2.x
Nom du paquet/de la classe	<code>com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider</code>
Création	<code>new WebIdentityFederationSessionCredentialsProvider</code>	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>

Changer de catégorie	1.x	2.x
Actualisation asynchrone	Comportement par défaut	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
Configuration	Paramètres du constructeur	Configurer une <code>AssumeRoleWithWebIdentityRequest</code> dans un <code>StsClient</code> et dans un générateur

Classes remplacées

Classe 1.x	Classes de remplacement 2.x
<code>com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code> et <code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
<code>com.amazonaws.services.s3.S3CredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code> et <code>software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider</code>

Classes supprimées

Classe 1.x
<code>com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider</code>
<code>com.amazonaws.auth.PropertiesFileCredentialsProvider</code>

Changements de région

Cette section décrit les modifications mises en œuvre dans la version AWS SDK for Java 2.x pour l'utilisation des Regions classes `Region` et.

Configuration de la région

- Certains AWS services ne disposent pas de points de terminaison spécifiques à une région. Lorsque vous utilisez ces services, vous devez définir la région comme `Region.AWS_GLOBAL` ou `Region.AWS_CN_GLOBAL`.

Exemple

```
Region region = Region.AWS_GLOBAL;
```

- Les classes `com.amazonaws.regions.Regions` et `com.amazonaws.regions.Region` sont désormais combinées en une seule classe `software.amazon.awssdk.regions.Region`.

Mappages de méthodes et de noms de classes

Les tableaux suivants répertorient les classes liées aux régions entre les versions 1.x et 2.x du AWS SDK for Java Vous pouvez créer une instance de ces classes à l'aide de la méthode `of()`.

Exemple

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

1.x Changements de méthode de classe `Regions`

1.x	2.x
<code>Regions.fromName</code>	<code>Region.of</code>
<code>Regions.getName</code>	<code>Region.id</code>
<code>Regions.getDescription</code>	<code>Region.metadata().description()</code>
<code>Regions.getCurrentRegion</code>	Non pris en charge
<code>Regions.DEFAULT_REGION</code>	Non pris en charge

1.x	2.x
<code>Regions.name</code>	<code>Region.id</code>

1.x Changements de méthode de classe Region

1.x	2.x
<code>Region.getName</code>	<code>Region.id</code>
<code>Region.hasHttpsEndpoint</code>	Non pris en charge
<code>Region.hasHttpEndpoint</code>	Non pris en charge
<code>Region.getAvailableEndpoints</code>	Non pris en charge
<code>Region.createClient</code>	Non pris en charge

RegionMetadata changements de méthode de classe

1.x	2.x
<code>RegionMetadata.getName</code>	<code>RegionMetadata.name</code>
<code>RegionMetadata.getDomain</code>	<code>RegionMetadata.domain</code>
<code>RegionMetadata.getPartition</code>	<code>RegionMetadata.partition</code>

ServiceMetadata changements de méthode de classe

1.x	2.x
<code>Region.getServiceEndpoint</code>	<code>ServiceMetadata.endpointFor(Region)</code>
<code>Region.isServiceSupported</code>	<code>ServiceMetadata.regions().contains(Region)</code>

Modifications des opérations, des demandes et des réponses

Dans la version 2.x du SDK pour Java, les demandes sont transmises à une opération cliente. Par exemple, `DynamoDbClient`'s `PutItemRequest` est passé à `DynamoDbClient.putItem` l'opération. Ces opérations renvoient une réponse du Service AWS, telle que `PutItemResponse` a.

La version 2.x du SDK pour Java présente les modifications suivantes par rapport à la version 1.x.

- Les opérations comportant plusieurs pages de réponse disposent désormais d'une `Paginator` méthode permettant d'itérer automatiquement tous les éléments de la réponse.
- Vous ne pouvez pas modifier les demandes et les réponses.
- Vous devez créer des demandes et des réponses avec une méthode de générateur statique au lieu d'un constructeur. Par exemple, la version 1.x `new PutItemRequest().withTableName(...)` existe désormais `PutItemRequest.builder().tableName(...).build()`.
- Les opérations prennent en charge un moyen court de créer des demandes : `dynamoDbClient.putItem(request -> request.tableName(...))`.

Opérations de streaming

Les opérations de streaming telles qu'Amazon S3 `getObject` et `putObject` les méthodes prennent désormais en charge les E/S non bloquantes. Par conséquent, les POJO de demande et de réponse ne prennent plus un `InputStream` comme paramètre. Au lieu de cela, pour les demandes synchrones, l'objet de demande accepte `RequestBody`, qui est un flux d'octets. L'équivalent asynchrone accepte un `AsyncRequestBody`.

Exemple du **putObject** fonctionnement d'Amazon S3 dans la version 1.x

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

Exemple du **putObject** fonctionnement d'Amazon S3 dans la version 2.x

```
s3client.putObject(PutObjectRequest.builder()
    .bucket(BUCKET)
    .key(KEY)
    .build(),
    RequestBody.of(Paths.get("myfile.in")));
```


En parallèle, un objet de réponse en streaming accepte un `ResponseTransformer` pour les clients synchrones et un `AsyncResponseTransformer` pour les clients asynchrones.

Exemple du **getObject** fonctionnement d'Amazon S3 dans la version 1.x

```
S3Object o = s3.getObject(bucket, key);
S3ObjectInputStream s3is = o.getObjectContent();
FileOutputStream fos = new FileOutputStream(new File(key));
```

Exemple du **getObject** fonctionnement d'Amazon S3 dans la version 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
    ResponseTransformer.toFile(Paths.get("key")));
```

Dans le SDK pour Java 2.x, les opérations de réponse en continu disposent `AsBytes` d'une méthode permettant de charger la réponse en mémoire et de simplifier les conversions de type courantes en mémoire.

Modifications des exceptions

Les noms des classes d'exception, leurs structures et leurs relations ont changé.

`software.amazon.awssdk.core.exception.SdkException` est la nouvelle `Exception` classe de base étendue par toutes les autres exceptions.

Ce tableau fait correspondre les modifications des noms des classes des exceptions.

1.x	2.x
<code>com.amazonaws.SdkBaseException</code> <code>com.amazonaws.AmazonClientException</code>	<code>software.amazon.awssdk.core.exception.SdkException</code>
<code>com.amazonaws.SdkClientException</code>	<code>software.amazon.awssdk.core.exception.SdkClientException</code>
<code>com.amazonaws.AmazonServiceException</code>	<code>software.amazon.awssdk.awscore.exception.AwsServiceException</code>

Le tableau suivant répertorie les méthodes relatives aux classes d'exceptions entre les versions 1.x et 2.x.

1.x	2.x
<code>AmazonServiceException.getRequestId</code>	<code>SdkServiceException.requestId</code>
<code>AmazonServiceException.getServiceName</code>	<code>AwsServiceException.awsErrorDetails().serviceName</code>
<code>AmazonServiceException.getErrorCode</code>	<code>AwsServiceException.awsErrorDetails().errorCode</code>
<code>AmazonServiceException.getErrorMessage</code>	<code>AwsServiceException.awsErrorDetails().errorMessage</code>
<code>AmazonServiceException.getStatusCode</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode</code>
<code>AmazonServiceException.getHttpHeaders</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().headers</code>
<code>AmazonServiceException.rawResponse</code>	<code>AwsServiceException.awsErrorDetails().rawResponse</code>

Changements de sérialisation

Les SDK pour Java v1.x et v2.x diffèrent dans la manière dont ils sérialisent les objets `List` pour demander des paramètres.

Le SDK pour Java 1.x ne sérialise pas une liste vide, alors que le SDK pour Java 2.x sérialise une liste vide en tant que paramètre vide.

Par exemple, considérez un service avec un `SampleOperation` qui prend un `SampleRequest`. `SampleRequest` accepte deux paramètres : le type `String` `str1` et le type `ListlistParam`, comme indiqué dans les exemples suivants.

Exemple ou **SampleOperation** en 1.x

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

La journalisation au niveau du fil indique que le `listParam` paramètre n'est pas sérialisé.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

Exemple ou **SampleOperation** en 2.x

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

La journalisation au niveau du fil indique que le `listParam` paramètre est sérialisé sans aucune valeur.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

Modifications spécifiques au service

Modifications apportées à Amazon S3

Le SDK for Java 2.x désactive l'accès anonyme par défaut. Par conséquent, vous devez activer l'accès anonyme en utilisant `AnonymousCredentialsProvider`.

Changements de nom d'opération

De nombreux noms d'opérations pour le Amazon S3 client ont changé dans la version AWS SDK for Java 2.x. Dans la version 1.x, le client Amazon S3 n'est pas généré directement à partir de l'API du service. Cela génère un discordance entre les opérations du kit SDK et l'API du service. Dans la version 2.x, nous générons désormais le client Amazon S3 pour qu'il soit plus cohérent avec l'API du service.

Le tableau suivant indique les noms des opérations dans les deux versions.

Noms des opérations Amazon S3

1.x	2.x
abortMultipartUpload	abortMultipartUpload
changeObjectStorageClass	copyObject
completeMultipartUpload	completeMultipartUpload
copyObject	copyObject
copyPart	uploadPartCopy
createBucket	createBucket
deleteBucket	deleteBucket
deleteBucketAnalyticsConfiguration	deleteBucketAnalyticsConfiguration
deleteBucketCrossOriginConfiguration	deleteBucketCors
deleteBucketEncryption	deleteBucketEncryption
deleteBucketInventoryConfiguration	deleteBucketInventoryConfiguration
deleteBucketLifecycleConfiguration	deleteBucketLifecycle
deleteBucketMetricsConfiguration	deleteBucketMetricsConfiguration
deleteBucketPolicy	deleteBucketPolicy
deleteBucketReplicationConfiguration	deleteBucketReplication
deleteBucketTaggingConfiguration	deleteBucketTagging
deleteBucketWebsiteConfiguration	deleteBucketWebsite

1.x	2.x
<code>deleteObject</code>	<code>deleteObject</code>
<code>deleteObjectTagging</code>	<code>deleteObjectTagging</code>
<code>deleteObjects</code>	<code>deleteObjects</code>
<code>deleteVersion</code>	<code>deleteObject</code>
<code>disableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>doesBucketExist</code>	<code>headBucket</code>
<code>doesBucketExistV2</code>	<code>headBucket</code>
<code>doesObjectExist</code>	<code>headObject</code>
<code>enableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>generatePresignedUrl</code>	S3Presigner
<code>getBucketAccelerateConfiguration</code>	<code>getBucketAccelerateConfiguration</code>
<code>getBucketAcl</code>	<code>getBucketAcl</code>
<code>getBucketAnalyticsConfiguration</code>	<code>getBucketAnalyticsConfiguration</code>
<code>getBucketCrossOriginConfiguration</code>	<code>getBucketCors</code>
<code>getBucketEncryption</code>	<code>getBucketEncryption</code>
<code>getBucketInventoryConfiguration</code>	<code>getBucketInventoryConfiguration</code>
<code>getBucketLifecycleConfiguration</code>	<code>getBucketLifecycle</code> ou <code>getBucketLifecycleConfiguration</code>
<code>getBucketLocation</code>	<code>getBucketLocation</code>
<code>getBucketLoggingConfiguration</code>	<code>getBucketLogging</code>

1.x	2.x
<code>getBucketMetricsConfiguration</code>	<code>getBucketMetricsConfiguration</code>
<code>getBucketNotificationConfiguration</code>	<code>getBucketNotification</code> ou <code>getBucketNotificationConfiguration</code>
<code>getBucketPolicy</code>	<code>getBucketPolicy</code>
<code>getBucketReplicationConfiguration</code>	<code>getBucketReplication</code>
<code>getBucketTaggingConfiguration</code>	<code>getBucketTagging</code>
<code>getBucketVersioningConfiguration</code>	<code>getBucketVersioning</code>
<code>getBucketWebsiteConfiguration</code>	<code>getBucketWebsite</code>
<code>getObject</code>	<code>getObject</code>
<code>getObjectAcl</code>	<code>getObjectAcl</code>
<code>getObjectAsString</code>	<code>getObjectAsBytes().asUtf8String</code>
<code>getObjectMetadata</code>	<code>headObject</code>
<code>getObjectTagging</code>	<code>getObjectTagging</code>
<code>getResourceUrl</code>	S3Utilities#getUrl
<code>getS3AccountOwner</code>	<code>listBuckets</code>
<code>getUrl</code>	S3Utilities#getUrl
<code>headBucket</code>	<code>headBucket</code>
<code>initiateMultipartUpload</code>	<code>createMultipartUpload</code>
<code>isRequesterPaysEnabled</code>	<code>getBucketRequestPayment</code>

1.x	2.x
<code>listBucketAnalyticsConfigurations</code>	<code>listBucketAnalyticsConfigurations</code>
<code>listBucketInventoryConfigurations</code>	<code>listBucketInventoryConfigurations</code>
<code>listBucketMetricsConfigurations</code>	<code>listBucketMetricsConfigurations</code>
<code>listBuckets</code>	<code>listBuckets</code>
<code>listMultipartUploads</code>	<code>listMultipartUploads</code>
<code>listNextBatchOfObjects</code>	<code>listObjectsV2Paginator</code>
<code>listNextBatchOfVersions</code>	<code>listObjectVersionsPaginator</code>
<code>listObjects</code>	<code>listObjects</code>
<code>listObjectsV2</code>	<code>listObjectsV2</code>
<code>listParts</code>	<code>listParts</code>
<code>listVersions</code>	<code>listObjectVersions</code>
<code>putObject</code>	<code>putObject</code>
<code>restoreObject</code>	<code>restoreObject</code>
<code>restoreObjectV2</code>	<code>restoreObject</code>
<code>selectObjectContent</code>	<code>selectObjectContent</code>
<code>setBucketAccelerateConfiguration</code>	<code>putBucketAccelerateConfiguration</code>
<code>setBucketAcl</code>	<code>putBucketAcl</code>
<code>setBucketAnalyticsConfiguration</code>	<code>putBucketAnalyticsConfiguration</code>

1.x	2.x
setBucketCrossOriginConfiguration	putBucketCors
setBucketEncryption	putBucketEncryption
setBucketInventoryConfiguration	putBucketInventoryConfiguration
setBucketLifecycleConfiguration	putBucketLifecycle ou putBucketLifecycleConfiguration
setBucketLoggingConfiguration	putBucketLogging
setBucketMetricsConfiguration	putBucketMetricsConfiguration
setBucketNotificationConfiguration	putBucketNotification ou putBucketNotificationConfiguration
setBucketPolicy	putBucketPolicy
setBucketReplicationConfiguration	putBucketReplication
setBucketTaggingConfiguration	putBucketTagging
setBucketVersioningConfiguration	putBucketVersioning
setBucketWebsiteConfiguration	putBucketWebsite
setObjectAcl	putObjectAcl
setObjectRedirectLocation	copyObject
setObjectTagging	putObjectTagging
uploadPart	uploadPart

Modifications apportées à Amazon SNS

Un client SNS ne peut plus accéder aux rubriques SNS dans des régions autres que celle à laquelle il est configuré pour accéder.

Modifications apportées à Amazon SQS

Un client SQS ne peut plus accéder aux files d'attente SQS dans des régions autres que celle à laquelle il est configuré pour accéder.

Modifications apportées à Amazon RDS

Le SDK pour Java 2.x l'`RdsUtilities#generateAuthenticationToken` utilise à la place de la `RdsIamAuthTokenGenerator` classe de la version 1.x.

Modifications du fichier de profil

Il AWS SDK for Java 2.x analyse les définitions de profil dans `~/.aws/config` et `~/.aws/credentials` pour émuler de plus près la façon dont la AWS CLI analyse les fichiers.

Le SDK pour Java 2.x :

- Résout un `~/` ou `~` suivi par le séparateur de chemin par défaut du système de fichiers au début du chemin en vérifiant, dans l'ordre `$HOME`, `$USERPROFILE` (Windows uniquement) `$HOMEDRIVE`, `$HOMEPATH` (Windows uniquement), puis la propriété du `user.home` système.
- Cherche la variable d'`AWS_SHARED_CREDENTIALS_FILE` environnement au lieu de `AWS_CREDENTIAL_PROFILES_FILE`.
- Supprime silencieusement les définitions de profil dans les fichiers de configuration sans le mot `profile` au début du nom du profil.
- Supprime silencieusement les définitions de profil qui ne sont pas composées de caractères alphanumériques, de traits de soulignement ou de tirets (une fois que le premier `profile` mot a été supprimé pour les fichiers de configuration).
- Fusionne les paramètres des définitions de profil dupliquées dans le même fichier.
- Fusionne les paramètres des définitions de profil dupliquées dans les fichiers de configuration et d'identification.
- Ne fusionne PAS les paramètres si `[profile foo]` `[foo]` les deux se trouvent dans le même fichier.

- Utilise les paramètres `[profile foo]` si `[profile foo]` les deux se `[foo]` trouvent dans le fichier de configuration.
- Utilise la valeur du dernier paramètre dupliqué dans le même fichier et le même profil.
- Reconnaît les deux `;` et `#` permet de définir un commentaire.
- Reconnaît `;` et permet de définir un commentaire `#` dans les définitions de profil, même si les caractères sont adjacents au crochet de fermeture.
- Reconnaît `;` et `#` définit un commentaire uniquement en définissant des valeurs uniquement si elles sont précédées d'un espace.
- Reconnaît `;` `#` et tous les contenus suivants lors de la définition des valeurs s'ils ne sont pas précédés d'espaces.
- Considère les informations d'identification basées sur les rôles comme les informations d'identification les plus prioritaires. Le SDK 2.x utilise toujours des informations d'identification basées sur les rôles si l'utilisateur spécifie la propriété. `role_arn`
- Considère les informations d'identification basées sur les sessions comme des informations d'identification. `second-highest-priority` Le SDK 2.x utilise toujours les informations d'identification basées sur les sessions si les informations d'identification basées sur les rôles n'ont pas été utilisées et que l'utilisateur spécifie les propriétés et. `aws_access_key_id` `aws_session_token`
- Utilise les informations d'identification de base si les informations d'identification basées sur les rôles et les sessions ne sont pas utilisées et si l'utilisateur a spécifié la propriété. `aws_access_key_id`

Variables d'environnement et modifications des propriétés du système

Variable d'environnement 1.x	Propriété du système 1.x	Variable d'environnement 2.x	Propriété du système 2.x
<code>AWS_ACCESS_KEY_ID</code> <code>AWS_ACCESS_KEY</code>	<code>aws.accessKeyId</code>	<code>AWS_ACCESS_KEY_ID</code>	<code>aws.accessKeyId</code>
<code>AWS_SECRET_KEY</code> <code>AWS_SECRET_ACCESS_KEY</code>	<code>aws.secretKey</code>	<code>AWS_SECRET_ACCESS_KEY</code>	<code>aws.secretAccessKey</code>

Variable d'environnement 1.x	Propriété du système 1.x	Variable d'environnement 2.x	Propriété du système 2.x
AWS_SESSION_TOKEN	<code>aws.sessionToken</code>	AWS_SESSION_TOKEN	<code>aws.sessionToken</code>
AWS_REGION	<code>aws.region</code>	AWS_REGION	<code>aws.region</code>
AWS_CONFIG_FILE		AWS_CONFIG_FILE	<code>aws.configFile</code>
AWS_CREDENTIALS_FILE		AWS_SHARED_CREDENTIALS_FILE	<code>aws.sharedCredentialsFile</code>
AWS_PROFILE	<code>aws.profile</code>	AWS_PROFILE	<code>aws.profile</code>
AWS_EC2_METADATA_DISABLED	<code>com.amazonaws.sdk.disableEc2Metadata</code>	AWS_EC2_METADATA_DISABLED	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	AWS_EC2_METADATA_SERVICE_ENDPOINT	<code>aws.ec2MetadataServiceEndpoint</code>
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI		AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	<code>aws.containerCredentialsPath</code>
AWS_CONTAINER_CREDENTIALS_FULL_URI		AWS_CONTAINER_CREDENTIALS_FULL_URI	<code>aws.containerCredentialsFullUri</code>

Variable d'environnement 1.x	Propriété du système 1.x	Variable d'environnement 2.x	Propriété du système 2.x
AWS_CONTAINER_AUTHORIZATION_TOKEN		AWS_CONTAINER_AUTHORIZATION_TOKEN	aws.containerAuthorizationToken
AWS_CBOR_DISABLED	com.amazonaws.sdk.disableCbor	CBOR_ENABLED	aws.cborEnabled
AWS_ION_BINARY_DISABLE	com.amazonaws.sdk.disableIoBinary	BINARY_ION_ENABLED	aws.binaryIonEnabled
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	aws.executionEnvironment
	com.amazonaws.sdk.disableCertChecking	Non pris en charge (fonctionnalité de demande)	Non pris en charge (fonctionnalité de demande)
	com.amazonaws.sdk.enableDefaultMetrics	Non pris en charge	Non pris en charge
	com.amazonaws.sdk.enableThrottledRetry	Non pris en charge	Non pris en charge

Variable d'environnement 1.x	Propriété du système 1.x	Variable d'environnement 2.x	Propriété du système 2.x
	<code>com.amazonaws.regions.RegionUtils.override</code>	Non pris en charge (fonctionnalité de demande)	Non pris en charge (fonctionnalité de demande)
	<code>com.amazonaws.regions.RegionUtils.disableRemote</code>	Non pris en charge (fonctionnalité de demande)	Non pris en charge (fonctionnalité de demande)
	<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>	Non pris en charge (fonctionnalité de demande)	Non pris en charge (fonctionnalité de demande)
	<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>	Non pris en charge (fonctionnalité de demande)	Non pris en charge (fonctionnalité de demande)

Changements dans Waiters de la version 1 à la version 2

Cette rubrique détaille les modifications apportées aux fonctionnalités de Waiters de la version 1 (v1) à la version 2 (v2).

Les tableaux suivants montrent la différence pour les serveurs DynamoDB en particulier. Les serveurs des autres services suivent le même schéma.

Changements de haut niveau

Les classes de serveurs utilisent le même artefact Maven que le service.

Modification	v1	v2
Dépendances de Maven	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.680¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>dynamodb</artif actId> </dependency> </dependencies> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.25.10²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>dynamodb</artif actId> </dependency> </dependencies> </pre>
Nom du package	com.amazonaws.serv ices.dynamodbv2.wa iters	software.amazon.aw ssdk.services.dyna modb.waiters

Modification	v1	v2
Noms des classes	AmazonDynamoDBWaiters	<ul style="list-style-type: none"> • Synchrones : DynamoDbWaiter • Asynchrones : DynamoDbAsyncWaiter

¹ [Dernière version.](#) ² [Dernière version.](#)

Modifications de l'API

Modification	v1	v2
Créez un serveur	<pre>AmazonDynamoDB client = AmazonDynamoDBClientBuilder .standard().build(); AmazonDynamoDBWaiters waiter = client.waiters();</pre>	<p>Synchrones :</p> <pre>DynamoDbClient client = DynamoDbClient.create(); DynamoDbWaiter waiter = client.waiter();</pre> <p>Asynchrones :</p> <pre>DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter waiter = asyncClient.waiter();</pre>
Patientez jusqu'à ce qu'une table existe	<p>Synchrones :</p> <pre>waiter.tableExists() .run(new WaiterParameters<>(</pre>	<p>Synchrones :</p> <pre>WaiterResponse<DescribeTableResponse> waiterResponse = waiter.waitUntilTableExists(</pre>

Modification	v1	v2
	<pre> new DescribeTableRequest(tableName)); </pre> <p>Asynchrone :</p> <pre> waiter.tableExists() .runAsync(new WaiterParameters() .withRequest(new DescribeTableRequest(tableName)), new WaiterHandler() { @Override public void onWaitSuccess(AmazonWebServiceRequest amazonWebServiceRequest) { System.out.println("Table creation succeeded"); } @Override public void onWaitFailure(Exception e) { e.printStackTrace(); } }).get(); </pre>	<pre> r -> r.tableName("myTable")); waiterResponse.matched().response() .ifPresent(t(System.out::println)); </pre> <p>Asynchrone :</p> <pre> waiter.waitUntilTableExists(r -> r.tableName(tableName)) .whenComplete((r, t) -> { if (t != null) { t.printStackTrace(); } else { System.out.println("Table creation succeeded"); } }).join(); </pre>

Configuration changes

Modification	v1	v2
Stratégie de sondage (nombre maximum de tentatives et délai fixe)	<pre> MaxAttemptsRetryStrategy maxAttemptsRetryStrategy = new MaxAttemptsRetryStrategy(10); FixedDelayStrategy fixedDelayStrategy = new FixedDelayStrategy(3); PollingStrategy pollingStrategy = new PollingStrategy(maxAttemptsRetryStrategy, fixedDelayStrategy); waiter.tableExists().run(new WaiterParameters<>(new DescribeTableRequest(tableName), pollingStrategy); </pre>	<pre> FixedDelayBackoffStrategy fixedDelayBackoffStrategy = FixedDelayBackoffStrategy.create(Duration.ofSeconds(3)); waiter.waitUntilTableExists(r -> r.tableName(tableName), c -> c.maxAttempts(10) .backoffStrategy(fixedDelayBackoffStrategy)); </pre>

Modifications apportées à Amazon S3 Transfer Manager de la version 1 à la version 2

Cette rubrique détaille les modifications apportées à Amazon S3 Transfer Manager de la version 1 (v1) à la version 2 (v2).

Changements de haut niveau

Modification	v1	v2
Dépendances de Maven	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-s3</ artifactId> </dependency> </dependencies> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifactId>s3- transfer-manager</art ifactId> </dependency> <dependency> <groupId> software.amazon.aw ssdk.crt</groupId> <artifact Id>aws-crt</artifa ctId> <version> 0.28.7³</version> </dependency> </dependencies> </pre>

Modification	v1	v2
Nom du package	com.amazonaws.serv ices.s3.transfer	software.amazon.aw ssdk.transfer.s3
Nom de classe	TransferManager	S3TransferManager

¹ [Dernière version.](#) ² [Dernière version.](#) ³ [Dernière version.](#)

Modifications de l'API de configuration

Paramètre	v1	v2
(trouvez un constructeur)	<pre>TransferManagerBuilder tmBuilder = TransferManagerBui lder.standard();</pre>	<pre>S3TransferManager. Builder tmBuilder = S3TransferManager. builder();</pre>
Client S3	<pre>tmBuilder.withS3Cl ient(...); tmBuilder.setS3C lient(...);</pre>	<pre>tmBuilder.s3Client (...);</pre>
Exécuteur	<pre>tmBuilder.withExec utorFactory(...); tmBuilder.setExecu torFactory(...);</pre>	<pre>tmBuilder.executor (...);</pre>
Arrêter les pools de threads	<pre>tmBuilder.withShut DownThreadPools(...); tmBuilder.setS hutdownThreadPools (...);</pre>	Non pris en charge. L'exécute ur fourni ne sera pas arrêté lorsque le S3 sera TransferM anager fermé
Taille minimale de la pièce à télécharger	<pre>tmBuilder.withMini mumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtB uilder().</pre>

Paramètre	v1	v2
	<pre>tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>minimumPartSizeInBytes(...) .build(); tmBuilder.s3Client(s3);</pre>
Seuil de téléchargement en plusieurs parties	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder() .thresholdInBytes(...) .build(); tmBuilder.s3Client(s3);</pre>
Taille minimale des parties de copie	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder() .minimumPartSizeInBytes(...) .build(); tmBuilder.s3Client(s3);</pre>
Seuil de copie en plusieurs parties	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder() .thresholdInBytes(...) .build(); tmBuilder.s3Client(s3);</pre>

Paramètre	v1	v2
Désactiver les téléchargements parallèles	<pre>tmBuilder.withDisableParallelDownloads(...); tmBuilder.setDisableParallelDownloads(...);</pre>	<p>Désactivez les téléchargements parallèles en transmettant un client S3 standard basé sur Java au gestionnaire de transfert.</p> <pre>S3AsyncClient s3 = S3AsyncClient.builder().build(); tmBuilder.s3Client(s3);</pre>
Calculez toujours le md5 en plusieurs parties	<pre>tmBuilder.withAlwaysCalculateMultipartMd5(...); tmBuilder.setAlwaysCalculateMultipartMd5(...);</pre>	Non pris en charge.

Changements de comportement


Le transfert parallèle nécessite un client AWS S3 basé sur CRT

[Dans le SDK pour Java 2.x, la fonctionnalité de transfert parallèle automatique \(chargement/téléchargement en plusieurs parties\) est disponible via le client S3 basé sur CRT.AWS](#) Pour activer la fonctionnalité de transfert parallèle, vous devez ajouter explicitement la dépendance à la [bibliothèque AWS Common Runtime \(CRT\)](#) afin d'optimiser les performances.

Le client S3 AWS basé sur CRT `S3TransferManager` permet à lui seul, sans utilisation, d'optimiser les performances des transferts parallèles. `S3TransferManagerLa v2` fournit des API supplémentaires qui facilitent le transfert de fichiers et de répertoires.

La capacité `S3TransferManager` à effectuer des transferts parallèles dépend de la manière dont `S3TransferManager` il est initié et de la déclaration de dépendance de la bibliothèque AWS Common Runtime (CRT).

Le tableau suivant décrit trois scénarios d'initialisation pour une `S3TransferManager` v2 avec et sans le AWS CRT déclaré comme dépendance.

Approche d'initialisation de S3 TransferManager v2	Le AWS CRT est-il déclaré comme dépendance ?	
	oui	non
<p>Initialiser le S3TransferManager sans transmettre d'instance S3AsyncClient</p> <p>Méthode de création statique :</p> <pre>S3TransferManager.create();</pre> <p>- OU -</p> <p>Méthode Builder :</p> <pre>S3TransferManager.builder().build();</pre>	 transfert parallèle automatique activé	 transfert parallèle automatique désactivé
<p>Transmettre une S3AsyncClient instance construite avec l'une des méthodes du générateur crt* ()</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p>- OU -</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	 transfert parallèle automatique activé	 erreur d'exécution
<p>Transmettez une S3AsyncClient instance créée avec l'une des méthodes de création standard afin que le gestionnaire de transfert n'ait aucune référence au CRT</p>		

Approche d'initialisation de S3 TransferManager v2	Le AWS CRT est-il déclaré comme dépendance ?	
<pre data-bbox="121 262 1015 451">S3AsyncClient s3AsyncClient = S3AsyncClient.builder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p data-bbox="121 493 203 525">- OU -</p> <pre data-bbox="121 567 1015 756">S3AsyncClient s3AsyncClient = S3AsyncClient.create(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	transfert parallèle automatique désactivé	transfert parallèle automatique désactivé

Téléchargement parallèle via des extractions par plage d'octets

Lorsque la fonction de transfert parallèle automatique est activée, le S3 Transfer Manager v2 utilise des extractions [par plage d'octets pour récupérer des](#) parties spécifiques de l'objet en parallèle (téléchargement en plusieurs parties). La façon dont un objet est téléchargé avec la version v2 ne dépend pas de la manière dont l'objet a été initialement chargé. Tous les téléchargements peuvent bénéficier d'un débit élevé et d'une simultanéité.

En revanche, avec S3 Transfer Manager v1, la manière dont l'objet a été initialement chargé est importante. Le S3 Transfer Manager v1 récupère les parties de l'objet de la même manière que les parties ont été téléchargées. Si un objet a été initialement chargé en tant qu'objet unique, le S3 Transfer Manager v1 n'est pas en mesure d'accélérer le processus de téléchargement en utilisant des sous-requêtes.

Comportement de défaillance

Avec S3 Transfer Manager v1, une demande de transfert de répertoire échoue si une sous-demande échoue. Contrairement à la v1, le futur renvoyé par S3 Transfer Manager v2 se termine avec succès même si certaines sous-requêtes échouent.

Par conséquent, vous devez vérifier l'absence d'erreurs dans la réponse en utilisant la [CompletedDirectoryDownload.failedTransfers\(\)](#) méthode ou [CompletedDirectoryUpload.failedTransfers\(\)](#) la méthode, même si le futur se termine avec succès.

Modifications apportées à l'utilitaire de métadonnées EC2 de la version 1 à la version 2

Cette rubrique décrit les modifications apportées à l'utilitaire de métadonnées Amazon Elastic Compute Cloud (EC2) du SDK pour Java, de la version 1 (v1) à la version 2 (v2).

Changements de haut niveau

Modification	v1	v2
Dépendances de Maven	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587<sup>1</sup></version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-co re</artifactId> </dependency> </dependencies> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21<sup>2</sup></version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>imds</artifactId> </dependency> <dependency> <groupId> software.amazon.aw ssdk</groupId> </pre>

Modification	v1	v2
		<pre><artifact Id>apache-client ³</ artifactId> </dependency> </dependencies></pre>
Nom du package	com.amazonaws.util	software.amazon.awssdk.imds
Approche d'instanciation	<p>Utilisez des méthodes utilitaires statiques ; pas d'instanciation :</p> <pre>String localHostName = EC2MetadataUtils.getLocalHostName();</pre>	<p>Utilisez une méthode d'usine statique :</p> <pre>Ec2MetadataClient client = Ec2MetadataClient.create();</pre> <p>Ou utilisez une approche de constructeur :</p> <pre>Ec2MetadataClient client = Ec2MetadataClient.builder() .endpointMode(EndpointMode.IPV6) .build();</pre>
Types de clients	Méthodes utilitaires synchrones uniquement : EC2MetadataUtils	<p>Synchrone : Ec2MetadataClient</p> <p>Asynchrone : Ec2MetadataAsyncClient</p>

¹ [Dernière version.](#) ² [Dernière version.](#)

³ Notez la déclaration du apache-client module pour la version 2. La version V2 de l'utilitaire de métadonnées EC2 nécessite une implémentation de l'SdkHttpClientinterface pour le client de

métadonnées synchrone ou de l'`SdkAsyncHttpClient` interface pour le client de métadonnées asynchrone. La [???](#) section présente la liste des clients HTTP que vous pouvez utiliser.

Demande de métadonnées

Dans la version 1, vous utilisez des méthodes statiques qui n'acceptent aucun paramètre pour demander des métadonnées pour une ressource EC2. En revanche, vous devez spécifier le chemin d'accès à la ressource EC2 en tant que paramètre dans la version v2. Le tableau suivant présente les différentes approches.

v1	v2
<pre>String userMetaData = EC2MetadataUtils.getUserData();</pre>	<pre>Ec2MetadataClient client = Ec2MetadataClient.create(); Ec2MetadataResponse response = client.get("/latest/user-data"); String userMetaData = response.asString();</pre>

Reportez-vous aux [catégories de métadonnées de l'instance](#) pour trouver le chemin que vous devez fournir pour demander un élément de métadonnées.

Note

Lorsque vous utilisez un client de métadonnées d'instance dans la version 2, vous devez vous efforcer d'utiliser le même client pour toutes les demandes de récupération de métadonnées.

Changements de comportement

Données JSON

Sur EC2, le service de métadonnées d'instance (IMDS) exécuté localement renvoie certaines métadonnées sous forme de chaînes au format JSON. Les métadonnées dynamiques d'un [document d'identité d'instance en sont un exemple](#).

L'API v1 contient des méthodes distinctes pour chaque élément de métadonnées d'identité d'instance, tandis que l'API v2 renvoie directement la chaîne JSON. Pour utiliser la chaîne JSON, vous pouvez utiliser l'[API Document](#) pour analyser la réponse et parcourir la structure JSON.

Le tableau suivant compare la manière dont vous récupérez les métadonnées d'un document d'identité d'instance dans les versions 1 et 2.

Cas d'utilisation	v1	v2
Récupérez la région	<pre>InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String region = instanceInfo.getRegion();</pre>	<pre>Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String region = instanceInfo.asMap().get("region").asString();</pre>
Récupérez l'identifiant de l'instance	<pre>InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String instanceId = instanceInfo.getInstanceId();</pre>	<pre>Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId = instanceInfo.asMap().get("instanceId").asString();</pre>
Récupérez le type d'instance	<pre>InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo();</pre>	<pre>Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document");</pre>

Cas d'utilisation	v1	v2
	<pre>String instanceType = instanceInfo.insta nceType();</pre>	<pre>Document instanceInfo = response.asDocumen t(); String instanceType = instanceInfo.asMap ().get("instanceTy pe").asString();</pre>

Différences de résolution des terminaux

Le tableau suivant indique les emplacements que le SDK vérifie pour convertir le point de terminaison en IMDS. Les emplacements sont répertoriés par ordre de priorité décroissant.

v1	v2
Propriété du système : <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	Méthode de configuration du générateur de clients : <code>endpoint(...)</code>
Variable d'environnement : <code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	Propriété du système : <code>aws.ec2MetadataServiceEndpoint</code>
Valeur par défaut: <code>http://169.254.169.254</code>	Fichier de configuration : <code>~.aws/config</code> avec le <code>ec2_metadata_service_endpoint</code> paramètre
	Valeur associée à la résolution <code>endpoint-mode</code>
	Valeur par défaut : <code>http://169.254.169.254</code>

Résolution du point de terminaison en version 2

Lorsque vous définissez explicitement un point de terminaison à l'aide du générateur, cette valeur de point de terminaison est prioritaire par rapport à tous les autres paramètres. Lorsque

le code suivant s'exécute, la propriété `aws.ec2MetadataServiceEndpoint` système et le `ec2_metadata_service_endpoint` paramètre du fichier de configuration sont ignorés s'ils existent.

```
Ec2MetadataClient client = Ec2MetadataClient
    .builder()
    .endpoint(URI.create("endpoint.to.use"))
    .build();
```

Mode Endpoint

Avec la version v2, vous pouvez spécifier un mode de point de terminaison pour configurer le client de métadonnées afin qu'il utilise les valeurs de point de terminaison par défaut pour IPv4 ou IPv6. Le mode Endpoint n'est pas disponible pour la version 1. La valeur par défaut utilisée pour IPv4 est `http://169.254.169.254` et `http://[fd00:ec2::254]` pour IPv6.

Le tableau suivant montre les différentes manières de définir le mode de point de terminaison par ordre décroissant de priorité.

		Valeurs possibles
Méthode de configuration du générateur de clients : <code>endpointMode(...)</code>	<pre>Ec2MetadataClient client = Ec2MetadataClient .builder() .endpointMode(EndpointMode.IPV4) .build();</pre>	<code>EndpointMode.IPV4</code> , <code>EndpointMode.IPV6</code>
Propriété du système	<code>aws.ec2MetadataServiceEndpointMode</code>	IPv4, IPv6 (le cas n'a pas d'importance)
Fichier de configuration : <code>~.aws/config</code>	Paramètre <code>ec2_metadata_service_endpoint</code>	IPv4, IPv6 (le cas n'a pas d'importance)
Non spécifié dans les méthodes précédentes	IPv4 est utilisé	

Comment le SDK se résout **endpoint** ou **endpoint-mode** dans la version v2

1. Le SDK utilise la valeur que vous avez définie dans le code du générateur de clients et ignore les paramètres externes. Étant donné que le SDK génère une exception si les deux `endpoint` et `endpointMode` sont appelés sur le générateur de clients, le SDK utilise la valeur du point de terminaison, quelle que soit la méthode que vous utilisez.
2. Si vous ne définissez aucune valeur dans le code, le SDK se tourne vers une configuration externe, d'abord pour les propriétés du système, puis pour un paramètre dans le fichier de configuration.
 - a. Le SDK vérifie d'abord la valeur d'un point de terminaison. Si une valeur est trouvée, elle est utilisée.
 - b. Si le SDK n'a toujours pas trouvé de valeur, il recherche les paramètres du mode endpoint.
3. Enfin, si le SDK ne trouve aucun paramètre externe et que vous n'avez pas configuré le client de métadonnées dans le code, le SDK utilise la valeur IPv4 de `http://169.254.169.254`

IMDSv2

Amazon EC2 définit deux approches pour accéder aux métadonnées des instances :

- Service de métadonnées d'instance, version 1 (IMDSv1) — Approche requête/réponse
- Service de métadonnées d'instance version 2 (IMDSv2) — Approche axée sur les sessions

Le tableau suivant compare le fonctionnement des SDK Java avec IMDS.

v1	v2
IMDSv2 est utilisé par défaut	Utilise toujours IMDSv2
Tente de récupérer un jeton de session pour chaque demande et revient à IMDSv1 s'il ne parvient pas à récupérer un jeton de session	Conserve un jeton de session dans un cache interne qui est réutilisé pour plusieurs demandes

Le SDK pour Java 2.x prend uniquement en charge IMDSv2 et ne revient pas à IMDSv1.

Différences de configuration

Le tableau suivant répertorie les différentes options de configuration.

Configuration	v1	v2
Nouvelle tentative	Configuration non disponible	Configurable via la méthode du générateur <code>retryPolicy(...)</code>
HTTP	Le délai d'expiration de la connexion est configurable via la variable d'AWS_METADATA_SERVICE_TIMEOUT environnement. La valeur par défaut est de 1 seconde.	Configuration disponible en passant un client HTTP à la méthode du générateur <code>rhttpClient(...)</code> . Le délai de connexion par défaut pour les clients HTTP est de 2 secondes.

Exemple de configuration HTTP v2

L'exemple suivant montre comment configurer le client de métadonnées. Cet exemple configure le délai d'expiration de la connexion et utilise le client HTTP Apache.

```

SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();

Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();

```

Modifications apportées à Amazon CloudFront Presign de la version 1 à la version 2

Cette rubrique détaille les modifications apportées à Amazon CloudFront entre la version 1 (v1) et la version 2 (v2).

Changements de haut niveau

Modification	v1	v2
Dépendances de Maven	<pre data-bbox="592 338 1027 1528"><dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587<sup>1</sup></version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>cloudfront</art ifactId> </dependency> </dependencies></pre>	<pre data-bbox="1070 338 1505 1493"><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21<sup>2</sup></version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>cloudfront</art ifactId> </dependency> </dependencies></pre>
Nom du package	com.amazonaws.services.cloudfront	software.amazon.awssdk.services.cloudfront
Noms des classes	CloudFrontUrlSigner CloudFrontCookieSigner	CloudFrontUtilities SignedUrl

Modification	v1	v2
		CannedSignerRequest CustomSignerRequest

¹ [Dernière version.](#) ² [Dernière version.](#)

Modifications de l'API

Attitude	v1	v2
Créez une demande prédéfinie	Les arguments sont transmis directement à l'API.	<pre>CannedSignerRequest cannedRequest = CannedSignerRequest.builder() .resourceUrl(resourceUrl) .privateKey(privateKey) .keyPairId(keyPairId) .expirationDate(expirationDate) .build();</pre>
Créez une demande personnalisée	Les arguments sont transmis directement à l'API.	<pre>CustomSignerRequest customRequest = CustomSignerRequest.builder() .resourceUrl(resourceUrl) .privateKey(keyFile)</pre>

Attitude	v1	v2
		<pre> .keyPairId(keyPairId) .expirationDate(ex pirationDate) .activeDate(active Date) .ipRange(ipRange) .build(); </pre>
Générer une URL signée (standardisée)	<pre> String signedUrl = CloudFrontUrlSigne r.getSignedURLWith CannedPolicy(resourceUrl, keyPairId, privateKe y, expirationDate); </pre>	<pre> CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilitie s.create(); SignedUrl signedUrl = cloudFrontUtilitie s.getSignedUrlWith CannedPolicy(canne dRequest); String url = signedUrl .url(); </pre>
Générer un cookie signé (personnalisé)	<pre> CookiesForCustomPolicy cookies = CloudFrontCookieSi gner.getCookiesFor CustomPolicy(resourceUrl, privateKey, keyPairId , expirationDate, activeDate, ipRange); </pre>	<pre> CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilitie s.create(); CookiesForCustomPolicy cookies = cloudFrontUtilitie s.getCookiesForCus tomPolicy(customRe quest); </pre>

En-têtes de cookies refactorisés dans la version 2

Dans Java v1, le SDK Java fournit les en-têtes de cookies sous forme de `Map.Entry<String, String>`

```
Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"
```

Le SDK Java v2 fournit l'intégralité de l'en-tête en une seule `String` fois.

```
String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-Signature=[SIGNATURE_VALUE]"
```

Modifications apportées à l'analyse des URI Amazon S3 de la version 1 à la version 2

Cette rubrique détaille les modifications apportées à l'analyse des URI Amazon S3 de la version 1 (v1) à la version 2 (v2.).

Changements de haut niveau

Pour commencer à analyser un URI S3 dans la version 1, vous instanciez un URI à l'aide `AmazonS3URI` d'un constructeur. Dans la v2, vous `parseUri()` faites appel à une instance de `S3Utilities`, pour renvoyer un `S3URI`.

Modification	v1	v2
Dépendances de Maven	<pre><dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587</version></pre>	<pre><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21</version></pre>

Modification	v1	v2
	<pre> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>s3</artifactId> </dependency> </dependencies> </pre>	<pre> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>s3</artifactId> </dependency> </dependencies> </pre>
Nom du package	com.amazonaws.serv ices.s3	software.amazon.aw ssdk.services.s3
Noms des classes	AmazonS3URI	S3URI

¹ [Dernière version.](#) ² [Dernière version.](#)

Modifications de l'API

Attitude	v1	v2
Analyse un URI S3.	<pre> URI uri = URI.creat e("https://s3.amazon aws.com"); AmazonS3Uri s3Uri = new AmazonS3U RI(uri, false); </pre>	<pre> S3Client s3Client = S3Client.create(); S3Utilities s3Utiliti es = s3Client.utilities (); S3Uri s3Uri = </pre>

Attitude	v1	v2
		<pre>s3Utilities.parseUri(uri);</pre>
Récupérez le nom du compartiment à partir d'une URI S3.	<pre>String bucket = s3Uri.getBucket();</pre>	<pre>Optional<String> bucket = s3Uri.bucket();</pre>
Récupérez la clé.	<pre>String key = s3Uri.getKey();</pre>	<pre>Optional<String> key = s3Uri.key();</pre>
Récupérez la région.	<pre>String region = s3Uri.getRegion();</pre>	<pre>Optional<Region> region = s3Uri.region(); String region; if (s3Uri.region().isPresent()) { region = s3Uri.region().get().id(); }</pre>
Vérifiez si l'URI S3 est un style de chemin.	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>
Récupérez l'ID de version.	<pre>String versionId = s3Uri.getVersionId();</pre>	<pre>Optional<String> versionId = s3Uri.firstMatchingRawQueryParameter("versionId");</pre>
Récupérez les paramètres de requête.	N/A	<pre>Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();</pre>

Changements de comportement

Codage d'URL

v1 fournit la possibilité de transmettre un indicateur pour spécifier si l'URI doit être codé en URL. La valeur par défaut est `true`.

Dans la version 2, le codage d'URL n'est pas pris en charge. Si vous travaillez avec des clés d'objet ou des paramètres de requête contenant des caractères réservés ou non sécurisés, vous devez les encoder par URL. Par exemple, vous devez remplacer un espace blanc " " par `%20`.

Modifications apportées à l'API IAM Policy Builder de la version 1 à la version 2

Cette rubrique décrit les modifications apportées à l'API IAM Policy Builder de la version 1 (v1) à la version 2 (v2).

Changements de haut niveau

Modification	v1	v2
Dépendances de Maven	<pre><dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency></pre>	<pre><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency></pre>

Modification	v1	v2
	<pre><groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-co re</artifactId> </dependency> </dependencies></pre>	<pre><groupId> software.amazon.aw ssdk</groupId> <artifact Id>iam-policy-buil der</artifactId> </dependency> </dependencies></pre>
Nom du package	com.amazonaws.auth.policy	software.amazon.awssdk.policybuilder.iam
Noms des classes	<p>Stratégie</p> <p>Instruction</p> <ul style="list-style-type: none"> • Déclaration. Effet • IdentityManagementActions • Ressource • Principal • Condition 	<p>IamPolicy</p> <p>IamStatement</p> <ul style="list-style-type: none"> • IamEffect • IamAction • IamResource • IamPrincipal • IamCondition • IamConditionOperator • IamConditionKey

¹ [Dernière version.](#) ² [Dernière version.](#)

Modifications de l'API

Paramètre	v1	v2
Instancier une politique	<pre>Policy policy = new Policy();</pre>	<pre>IamPolicy.Builder policyBuilder = IamPolicy.builder(); ...</pre>

Paramètre	v1	v2
		<pre>IamPolicy policy = policyBuilder.build();</pre>
Définir l'identifiant	<pre>policy.withId(...); policy.setId(...);</pre>	<pre>policyBuilder.id(...);</pre>
Version du set	N/A - utilise la version par défaut de 2012-10-17	<pre>policyBuilder.version(...);</pre>
Créer une déclaration	<pre>Statement statement = new Statement (Effect.Allow) .withActions(...) .withConditions(...) .withId(...) .withPrincipals(...) .withResources(...);</pre>	<pre>IamStatement statement = IamStatement.builder() .effect(IamEffect.ALLOW) .actions(...) .notActions(...) .conditions(...) .sid(...) .principals(...) .notPrincipals(...) .resources(...) .notResources(...) .build();</pre>
Définir une déclaration	<pre>policy.withStatements(statement); policy.setStatements(statement);</pre>	<pre>policyBuilder.addStatement(statement);</pre>

Différences dans l'élaboration d'une déclaration

Actions

v1

Le SDK v1 comporte des [enumtypes d'actions](#) de service qui représentent des [Action](#) éléments d'une déclaration de politique. Les enum types suivants en sont des exemples.

- [IdentityManagementActions](#)
- [DynamoDBv2Actions](#)
- [SQSActions](#)

L'exemple suivant montre la `SendMessage` constante pour `SQSActions`.

```
Action action = SQSActions.SendMessage;
```

Vous ne pouvez pas spécifier d'[NotAction](#) élément dans une instruction dans la version 1.

v2

Dans la version 2, l'[IamAction](#) interface représente toutes les actions. Pour spécifier un élément d'[action spécifique au service](#), transmettez une chaîne à la `create` méthode comme indiqué dans le code suivant.

```
IamAction action = IamAction.create("sqs:SendMessage");
```

Vous pouvez spécifier a [NotAction](#) pour une instruction avec v2 comme indiqué dans le code suivant.

```
IamAction action = IamAction.create("sqs:SendMessage");  
IamStatement.builder().addNotAction(action);
```

Conditions

v1

Pour représenter les conditions des instructions, le SDK v1 utilise des sous-classes de [Condition](#)

- [ArnCondition](#)

- [BooleanCondition](#)
- [DateCondition](#)
- [IpAddressCondition](#)
- [NumericCondition](#)
- [StringCondition](#)

Chaque `Condition` sous-classe définit un enum type de comparaison pour aider à définir la condition. Par exemple, ce qui suit montre une comparaison de [chaînes différentes](#) pour une condition.

```
Condition condition = new StringCondition(StringComparisonType.StringNotLike, "key", "value");
```

v2

Dans la version 2, vous créez une condition pour une déclaration de politique en utilisant [IamCondition](#) et en fournissant un [IamConditionOperator](#), qui contient enums pour tous les types.

```
IamCondition condition = IamCondition.create(IamConditionOperator.STRING_NOT_LIKE, "key", "value");
```

Ressources

v1

L'[Resource](#) élément d'une déclaration de politique est représenté par la [Resource](#) classe du SDK. Vous fournissez l'ARN sous forme de chaîne dans le constructeur. Les sous-classes suivantes fournissent des constructeurs pratiques.

- [S3 BucketResource](#)
- [S3 ObjectResource](#)
- [SQS QueueResource](#)

Dans la version 1, vous pouvez spécifier un [NotResource](#) élément pour a [Resource](#) en appelant la `withIsNotType` méthode comme indiqué dans l'instruction suivante.

```
Resource resource = new Resource("arn:aws:s3:::mybucket").withIsNotType(true);
```

v2

Dans la version 2, vous créez un [Resource](#) élément en transmettant un ARN à la `IamResource.create` méthode.

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
```

Un [IamResource](#) peut être défini comme [NotResource](#) élément comme indiqué dans l'extrait suivant.

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");  
IamStatement.builder().addNotResource(resource);
```

`IamResource.ALL` représente toutes les ressources.

Principaux

v1

Le SDK v1 propose les [Principal](#) classes suivantes pour représenter les types de principes qui incluent tous les membres :

- `AllUsers`
- `AllServices`
- `AllWebProviders`
- `All`

Vous ne pouvez pas ajouter d'[NotPrincipal](#) élément à une déclaration.

v2

Dans la version 2, `IamPrincipal.ALL` représente tous les principes :

Pour représenter tous les membres dans d'autres types de directeurs, utilisez les [IamPrincipalType](#) classes lorsque vous créez un `IamPrincipal`.

- `IamPrincipal.create(IamPrincipalType.AWS, "*")` pour tous les utilisateurs.

- `IamPrincipal.create(IamPrincipalType.SERVICE, "*")` pour tous les services.
- `IamPrincipal.create(IamPrincipalType.FEDERATED, "*")` pour tous les fournisseurs Web.
- `IamPrincipal.create(IamPrincipalType.CANONICAL_USER, "*")` pour tous les utilisateurs canoniques.

Vous pouvez utiliser `addNotPrincipal` cette méthode pour représenter un [NotPrincipal](#) élément lorsque vous créez une déclaration de politique, comme indiqué dans l'instruction suivante.

```
IamPrincipal principal = IamPrincipal.create(IamPrincipalType.AWS,
    "arn:aws:iam::444455556666:root");
IamStatement.builder().addNotPrincipal(principal);
```

Utilisation du kit SDK for Java 1.x side-by-side

Vous pouvez utiliser les deux versions de AWS SDK for Java dans vos projets.

Vous trouverez ci-dessous un exemple de `pom.xml` fichier pour un projet utilisant Amazon S3 la version 1.x et DynamoDB la version 2.16.1.

Exemple Exemple de POM

Cet exemple montre une entrée de `pom.xml` fichier pour un projet qui utilise à la fois les versions 1.x et 2.x du SDK.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.12.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
```

```
        <scope>import</scope>
    </dependency>
</dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```

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.

Ce guide a été publié pour la dernière fois le 21 mai 2024.

Modification	Description	Date
Comment configurer le JVM TTL	Supprimez les instructions pour définir <code>networkaddress.cache.ttl</code> la propriété de sécurité à l'aide d'une propriété système de ligne de commande Java.	21 mai 2024
the section called “Réduisez le temps de démarrage du SDK pour AWS Lambda”	Mettre à jour les recommandations du client HTTP afin de réduire le temps de démarrage pour AWS Lambda	14 mai 2024
the section called “Mesures relatives aux clients du service”	Réorganiser les éléments du tableau des mesures	1er mai 2024
the section called “Résolution des problèmes”	Ajoutez une rubrique de résolution des problèmes.	26 avril 2024
the section called “Métriques collectées à chaque demande”	Ajoutez de nouvelles métriques signalées par le SDK.	26 avril 2024
the section called “Définissez le TTL de la JVM pour les recherches de noms DNS”	Remplacez le TTL de recherche DNS recommandé par 5 secondes.	23 avril 2024

Modification	Description	Date
the section called “Nom du package vers les mappages ArtifactID”	Ajoutez le nom du package à la rubrique de mappage ArtifactID de Maven.	17 avril 2024
the section called “Utiliser les métriques du SDK”	Ajoutez les détails de configuration dans la section des métriques.	12 avril 2024
the section called “API IAM Policy Builder”	Ajoutez les informations de migration de l'API IAM Policy Builder.	11 avril 2024
???	Mettez à jour les informations du proxy HTTP.	3 avril 2024
the section called “En toute sécurité”	Ajoutez des instructions pour désactiver IMDSv1.	14 mars 2024
the section called “tep-by-step Instructions S”	Ajoutez des instructions de step-by-step migration.	8 mars 2024
Migrer vers la version 2	Mettre à jour le sujet de migration.	14 février 2024
the section called “Configuration de clients AWS HTTP basés sur CRT”	Ajoutez des informations sur le client HTTP synchrone AWS basé sur CRT.	5 janvier 2024
the section called “Amazon Cognito Identity” et the section called “Fournisseur d'identité Amazon Cognito”	Les exemples Amazon Cognito ont été déplacés vers la section Exemples de code.	28 décembre 2023
Utiliser les fonctionnalités du SDK	La rubrique relative aux fonctionnalités du SDK a été retravaillée.	11 décembre 2023

Modification	Description	Date
Clé OpenPGP	Fournissez la clé OpenPGP actuelle.	6 décembre 2023
the section called “Changements de sérialisation”	Décrivez les différences de sérialisation entre la v1 et la v2 du SDK for Java.	5 décembre 2023
the section called “Gestionnaire de transfert S3”	Ajoutez une section qui détaille les modifications apportées au gestionnaire de transfert S3 de la version 1 à la version 2.	13 novembre 2023
the section called “Référence d'annotation”	Ajoutez une liste des annotations de classe de données qui peuvent être utilisées avec le client DynamoDB amélioré.	30 octobre 2023
???	Ajoutez des informations sur l'état de migration des bibliothèques et des utilitaires du SDK for Java v1.x vers la version 2.x	17 octobre 2023
???	Mettre à jour la rubrique de configuration de Gradle	17 octobre 2023
the section called “Ignorer les attributs nuls des objets imbriqués”	Ajoutez des informations sur l'annotation DynamoDB Enhanced Client. <code>@DynamoDbIgnoreNulls</code>	22 septembre 2023
the section called “Accès interrégional”	Ajoutez des informations sur l'accès interrégional aux compartiments Amazon S3.	31 août 2023

Modification	Description	Date
the section called “Préserver les objets vides”	Ajoutez une section qui traite de l' <code>@DynamoDbPreserveEmptyObject</code> annotation.	25 août 2023
???	Mettre à jour la section client du service.	15 août 2023
the section called “Recommandations des clients”	Depuis la version 0.23, AWS CRT prend en charge les systèmes d'exploitation basés sur Musl tels qu'Alpine Linux. Les recommandations du client HTTP reflètent désormais le support de musl.	11 août 2023
the section called “Création de politiques IAM”	Ajouter une section API IAM Policy Builder	31 juillet 2023
the section called “Mise en route”	Corrigez plusieurs extraits de code dans la section Get Started de la rubrique DynamoDB Enhanced Client.	24 juillet 2023
the section called “Configuration des proxys HTTP”	Ajoutez des informations de support relatives au proxy HTTP et des exemples pour chaque client HTTP.	2 juin 2023
Réorganiser la table des matières	Promouvez Exemples de code la section et les Travaillez avec Services AWS entrées de table des matières de haut niveau.	24 mai 2023

Modification	Description	Date
the section called “Ajouter une dépendance à la journalisation”	Afficher les dépendances de Gradle dans la section de journalisation.	23 mai 2023
the section called “Travailler avec des résultats paginés”	Mettre à jour la rubrique de pagination.	18 mai 2023
the section called “Configurer un projet Gradle”	Mettez à jour la configuration du projet Gradle.	3 mai 2023
API client améliorée DynamoDB	Publication d'une rubrique réécrite sur l'API client améliorée DynamoDB.	28 avril 2023
Mettre à jour les instructions du didacticiel de démarrage	Archétype Maven modifié pour inclure une option pour CredentialsProvider ; instructions modifiées en conséquence.	11 avril 2023
the section called “Recommandations des clients”	Ajouter un guide de décision pour les clients HTTP	30 mars 2023
Mises à jour des bonnes pratiques IAM	Mise à jour du guide s'aligner sur les bonnes pratiques IAM. Pour plus d'informations, consultez Bonnes pratiques de sécurité dans IAM .	14 mars 2023
the section called “Recharger les informations d'identification du profil”	Ajoutez une section sur le rechargement des informations d'identification du profil.	9 février 2023
the section called “Configuration de clients AWS HTTP basés sur CRT”	Mettre à jour le sujet pour la version GA.	8 février 2023

Modification	Description	Date
the section called “Utiliser les métadonnées d'instance Amazon EC2”	Ajoutez un exemple guidé pour le client Java SDK pour le service de métadonnées d'instance Amazon S3.	1er février 2023
the section called “Utilisez un client S3 performant”	Ajoutez une section pour le client S3 AWS basé sur CRT.	19 décembre 2022
the section called “Transférer des fichiers et des répertoires”	Mettez à jour les exemples d'Amazon S3 Transfer Manager pour la version GA.	19 décembre 2022
the section called “Bonnes pratiques”	Ajout d'une section sur les meilleures pratiques.	18 novembre 2022
the section called “Charger des informations d'identification temporaires à partir d'un processus externe”	Ajout d'une section sur le chargement des informations d'identification à partir d'un processus externe.	15 novembre 2022
the section called “Mesures relatives aux clients du service”	Liste des métriques mise à jour avec les exigences d'utilisation du client HTTP.	9 novembre 2022
the section called “Transférer des fichiers et des répertoires”	Exemple de code corrigé.	2 novembre 2022
the section called “Réduisez le temps de démarrage du SDK pour AWS Lambda”	Section mise à jour avec des options supplémentaires pour réduire le temps de démarrage de Lambda.	1er novembre 2022
the section called “Clients HTTP”	Des informations de configuration ont été ajoutées pour couvrir tous les clients HTTP du SDK.	26 octobre 2022

Modification	Description	Date
the section called “Journalisation”	Rubrique de journalisation mise à jour pour inclure les détails de journalisation des connexions pour tous les clients HTTP.	4 octobre 2022
the section called “AWS services de base de données”	Ajout d'une section de présentation des services de AWS base de données et du SDK pour Java 2.x.	13 septembre 2022
La mise en réseau EC2-Classic prend sa retraite	EC2-Classic prendra sa retraite le 15 août 2022.	28 juillet 2022
the section called “Options d'authentification supplémentaires”	Mise à jour de la dépendance requise pour l'authentification unique.	18 juillet 2022
the section called “protocole TLS (Transport Layer Security)”	Mettez à jour les informations de sécurité TLS.	8 avril 2022
the section called “Options d'authentification supplémentaires”	Ajout d'informations supplémentaires sur la configuration et l'utilisation des informations d'identification.	22 février 2021
the section called “Configuration d'un projet GraalVM Native Image”	Nouveau sujet pour configurer un projet GraalVM Native Image.	18 février 2021
the section called “Sondage sur l'état des ressources”	Les serveurs ont été publiés ; ajout d'un sujet pour la nouvelle fonctionnalité.	30 septembre 2020

Modification	Description	Date
the section called “Utiliser les métriques du SDK”	Métriques publiées ; ajout d'un sujet pour la nouvelle fonctionnalité.	17 août 2020
the section called “Amazon SNS”	Ajout d'exemples de sujets pour Amazon SNS.	30 mai 2020
the section called “Réduisez le temps de démarrage du SDK pour AWS Lambda”	Ajout d'une rubrique sur les performances des AWS Lambda fonctions.	29 mai 2020
the section called “Définissez le TTL de la JVM pour les recherches de noms DNS”	Ajout d'une rubrique sur la mise en cache DNS JVM TTL.	27 avril 2020
the section called “Configuration d'un projet Apache Maven”, the section called “Configurer un projet Gradle”	Nouveaux sujets de configuration pour Maven et Gradle.	21 avril 2020
the section called “protocole TLS (Transport Layer Security)”	Ajout de TLS 1.2 à la section sur la sécurité.	19 mars 2020
the section called “S'abonner à Amazon Kinesis Data Streams”	Exemples de Kinesis flux ajoutés.	2 août 2018
the section called “Travailler avec des résultats paginés”	Ajout d'une rubrique de pagination automatique.	5 avril 2018
???	Des exemples de sujets ont été ajoutés pour IAM Amazon EC2, CloudWatch et DynamoDB.	29 décembre 2017

Modification	Description	Date
the section called “Amazon S3”	Exemple de getObject ajouté pour. Amazon S3	7 août 2017
the section called “Utiliser la programmation asynchrone”	Ajout d'une rubrique asynchrone.	4 août 2017
Version GA de la version AWS SDK for Java 2.x	AWS SDK for Java version 2 (v2) publiée.	28 juin 2017

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.