

Guide du développeur

Kit AWS SDK pour Kotlin



Kit AWS SDK pour Kotlin: Guide du développeur

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

Les marques commerciales et la présentation commerciale d'Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d'une manière susceptible d'entraîner une confusion chez 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

Qu'est-ce que c'est Kit AWS SDK pour Kotlin ?	1
Premiers pas avec SDK	1
Maintenance et support pour les versions SDK majeures	1
Ressources supplémentaires	1
Mise en route	3
Étape 1 : Configuration de ce didacticiel	3
Étape 2 : Création du projet	3
Étape 3 : Écrire le code	6
Étape 4 : créer et exécuter l'application	8
Réussite	9
Nettoyage	9
Étapes suivantes	9
Configuration	10
Configuration de base	10
Présentation	10
Possibilité de connexion au portail d' AWS accès	12
Configuration de l'authentification unique	12
Connectez-vous à l'aide du AWS CLI	13
Installation de Java et d'un outil de compilation	14
Utiliser des informations d'identification temporaires	14
Création de fichiers de construction de projets	15
Codez votre projet	21
Connectez-vous à l'aide du AWS CLI	21
Configuration	22
Création d'un client de service	24
Configurer un client dans le code	24
Configuration d'un client depuis l'environnement	25
Fermez le client	26
Région AWS sélection	27
Chaîne de fournisseurs de régions par défaut	27
Fournisseurs d'informations d'identification	28
Chaîne de fournisseurs d'informations d'identification par défaut	28
Fournisseur d'informations d'identification explicite	31
Points de terminaison clients	32

Configuration personnalisée	32
Exemples	35
HTTP	37
Configuration du client HTTP	37
Utiliser un proxy HTTP	40
Intercepteurs	41
Appliquer une version minimale de TLS	43
Nouvelle tentative	44
Configuration par défaut	44
Nombre maximum de tentatives	45
Retards et retards	45
Réessayez le bucket de jetons	48
Tentatives adaptatives	51
Observabilité	53
Configurez un TelemetryProvider	53
Métriques	55
Journalisation	57
Fournisseurs de services de télémétrie	61
Remplacer la configuration du client	63
Cycle de vie du client annulé	63
Ressources partagées	64
Utilisation de la SDK	66
Faites des demandes	66
DSL Surcharges de l'interface de service	67
Demandes sans saisie requise	68
Coroutines	68
Faire des demandes simultanées	68
Faire des demandes de blocage	69
Opérations de streaming	70
Réponses en streaming	70
Demandes de streaming	71
Pagination	72
Programmes d'attente	73
Gestion des erreurs	74
Exceptions de service	74
Exceptions relatives aux	75

Métadonnées d'erreur	75
Demandes de pré-signature	76
Principes de base de la présignature	76
Configuration avancée de présignature	77
Présignature POST et demandes PUT	77
Opérations qu'ils SDK peuvent présigner	78
Résolution des problèmes de FAQs	79
Comment résoudre les problèmes de « fermeture de la connexion » ?	79
Pourquoi des exceptions sont-elles émises avant que le nombre maximum de tentatives n'ait été atteint ?	80
Comment puis-je réparer NoSuchMethodError ou NoClassDefFoundError ?	81
Travaillez avec Services AWS	84
Amazon S3	85
Protection de l'intégrité des données avec des checksums	86
Travaillez avec des points d'accès multirégionaux	90
DynamoDB	96
Utiliser des points de AWS terminaison basés sur des comptes	96
Utiliser DynamoDB Mapper (version préliminaire pour les développeurs)	97
Exemples de code	124
API Gateway	125
Scénarios	126
Aurora	126
Principes de base	127
Actions	140
Scénarios	126
Auto Scaling	154
Principes de base	127
Actions	140
Amazon Bedrock	172
Actions	140
CloudWatch	173
Principes de base	127
Actions	140
CloudWatch Journaux	214
Actions	140
Fournisseur d'identité Amazon Cognito	217

Actions	140
Scénarios	126
Amazon Comprehend	233
Scénarios	126
DynamoDB	234
Principes de base	127
Actions	140
Scénarios	126
Amazon EC2	265
Principes de base	127
Actions	140
Amazon ECR	295
Principes de base	127
Actions	140
OpenSearch Service	325
Actions	140
EventBridge	329
Principes de base	127
Actions	140
AWS Glue	360
Principes de base	127
Actions	140
IAM	371
Principes de base	127
Actions	140
AWS IoT	391
Principes de base	127
Actions	140
AWS IoT data	415
Actions	140
Amazon Keyspaces	417
Principes de base	127
Actions	140
AWS KMS	443
Actions	140
Lambda	452

Principes de base	127
Actions	140
Scénarios	126
MediaConvert	462
Actions	140
Amazon Pinpoint	476
Actions	140
Amazon RDS	486
Principes de base	127
Actions	140
Scénarios	126
Services de données Amazon RDS	504
Scénarios	126
Amazon Redshift	505
Actions	140
Scénarios	126
Amazon Rekognition	510
Actions	140
Scénarios	126
Enregistrement de domaine Route 53	528
Principes de base	127
Actions	140
Amazon S3	547
Principes de base	127
Actions	140
Scénarios	126
SageMaker IA	570
Actions	140
Scénarios	126
Secrets Manager	595
Actions	140
Amazon SES	596
Scénarios	126
Amazon SNS	598
Actions	140
Scénarios	126

Amazon SQS	626
Actions	140
Scénarios	126
Step Functions	649
Principes de base	127
Actions	140
Support	670
Principes de base	127
Actions	140
Amazon Translate	688
Scénarios	126
Sécurité	690
Protection des données	690
Application de TLS 1.2	692
Prise en charge de TLS dans Java	692
Comment vérifier la version de TLS	692
Gestion de l'identité et des accès	692
Public ciblé	693
Authentification par des identités	693
Gestion des accès à l'aide de politiques	697
Comment Services AWS travailler avec IAM	700
Résolution des problèmes AWS d'identité et d'accès	700
Validation de la conformité	702
Résilience	704
Sécurité de l'infrastructure	704
Historique du document	706
.....	dccx

Qu'est-ce que c'est Kit AWS SDK pour Kotlin ?

Kit AWS SDK pour Kotlin Il fournit Kotlin APIs pour Amazon Web Services. À l'aide de SDK, vous pouvez créer des applications Kotlin qui fonctionnent avec Amazon S3, AmazonEC2, Amazon DynamoDB, etc. Avec le KotlinSDK, vous pouvez cibler la JVM plateforme ou le API niveau Android 24 ou supérieur. Support pour des plateformes supplémentaires telles que JavaScript et Native dans les futures versions.

Pour suivre les fonctionnalités à venir dans les prochaines versions, consultez notre [feuille de route sur GitHub](#).

Premiers pas avec SDK

Pour commencer à utiliser le SDK, suivez le [Mise en route](#) didacticiel.

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

Pour créer et configurer des clients de service auxquels envoyer des demandes Services AWS, reportez-vous à [la section Configuration](#). Pour plus d'informations sur les différentes fonctionnalités du SDK, voir [Utilisation de la SDK](#).

Pour des cas d'utilisation et des exemples d'exécution d'API opérations spécifiques, voir [Exemples de code](#).

Maintenance et support pour les versions SDK majeures

Pour plus d'informations sur la maintenance et le support des versions SDK majeures et de leurs dépendances sous-jacentes, consultez les rubriques suivantes du guide de référence AWS SDKs et des outils :

- [AWS SDKset politique de maintenance des outils](#)
- [AWS SDKsMatrice de support des versions et outils](#)

Ressources supplémentaires

En plus de ce guide, les ressources en ligne suivantes sont de précieuses ressources en ligne SDK pour les développeurs de Kotlin :

- [AWS blog pour développeurs](#)
- [Forums pour développeurs](#)
- [SDKsource](#) (GitHub)
- [Catalogue d'exemples de code AWS](#)
- [@awsdevelopers](#) (X, anciennement Twitter)

Commencez avec le SDK pour Kotlin

Kit AWS SDK pour Kotlin Il fournit Kotlin APIs pour chacun Service AWS. À l'aide du SDK, vous pouvez créer des applications Kotlin qui fonctionnent avec Amazon S3 EC2, Amazon, Amazon DynamoDB, etc.

Ce didacticiel vous montre comment utiliser Gradle pour définir les dépendances pour. Kit AWS SDK pour Kotlin Vous créez ensuite du code qui écrit des données dans une table DynamoDB. Bien que vous souhaitiez peut-être utiliser les fonctionnalités d'un IDE, vous n'avez besoin pour ce didacticiel que d'une fenêtre de terminal et d'un éditeur de texte.

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 avez besoin d'un [ensemble d'autorisations IAM Identity Center](#) pouvant accéder à DynamoDB et d'un environnement de développement Kotlin configuré avec les paramètres d'authentification unique d'IAM Identity Center pour y accéder. AWS

Suivez les instructions [Configuration de base](#) de ce guide pour obtenir les instructions de base de ce didacticiel.

Une fois que vous avez configuré votre environnement de développement avec [un accès par authentification unique](#) pour le SDK Kotlin et que vous disposez d'une [session de portail d' AWS accès active](#), passez à l'étape 2.

Étape 2 : Création du projet

Pour créer le projet de ce didacticiel, utilisez d'abord Gradle pour créer les fichiers de base d'un projet Kotlin. Mettez ensuite à jour les fichiers avec les paramètres, les dépendances et le code requis pour le Kit AWS SDK pour Kotlin.

Pour créer un nouveau projet à l'aide de Gradle

Note

Ce didacticiel utilise la version 8.11.1 de Gradle avec la `gradle init` commande, qui propose cinq instructions à l'étape 3 ci-dessous. Si vous utilisez une version différente de Gradle, les instructions peuvent différer, de même que les versions préremplies des artefacts.

1. Créez un nouveau répertoire appelé `getstarted` à l'emplacement de votre choix, tel que votre bureau ou votre dossier personnel.
2. Ouvrez un terminal ou une fenêtre d'invite de commande et naviguez jusqu'au `getstarted` répertoire que vous avez créé.
3. Utilisez la commande suivante pour créer un nouveau projet Gradle et une classe Kotlin de base.

```
gradle init --type kotlin-application --dsl kotlin
```

- Lorsque vous êtes invité à saisir la cible `Java version`, appuyez sur `Enter` (par défaut sur `21`).
- Lorsque vous y êtes invité `Project name`, appuyez sur `Enter` (par défaut, le nom du répertoire, `getstarted` dans ce didacticiel).
- Lorsque vous y êtes invité `application structure`, appuyez sur `Enter` (par défaut `Single application project`).
- Lorsque vous y êtes invité `Select test framework`, appuyez sur `Enter` (par défaut `kotlin.test`).
- Lorsque vous y êtes invité `Generate build using new APIs and behavior`, appuyez sur `Enter` (par défaut `no`).

Pour configurer votre projet avec des dépendances pour Amazon S3 Kit AWS SDK pour Kotlin et Amazon S3

- Dans le `getstarted` répertoire que vous avez créé lors de la procédure précédente, remplacez le contenu du `settings.gradle.kts` fichier par le contenu suivant, en le `X.Y.Z` remplaçant par la [dernière version](#) du SDK pour Kotlin :

```
dependencyResolutionManagement {  
    repositories {
```

```
mavenCentral()
}

versionCatalogs {
    create("awssdk") {
        from("aws.sdk.kotlin:version-catalog:X.Y.Z")
    }
}

plugins {
    // Apply the foojay-resolver plugin to allow automatic download of JDKs.
    id("org.gradle.toolchains.foojay-resolver-convention") version "0.8.0"
}

rootProject.name = "getstarted"
include("app")
```

- Accédez au gradle répertoire situé à l'intérieur du getstarted répertoire. Remplacez le contenu du fichier de catalogue de versions nommé `libs.versions.toml` par le contenu suivant :

```
[versions]
junit-jupiter-engine = "5.10.3"

[libraries]
junit-jupiter-engine = { module = "org.junit.jupiter:junit-jupiter-engine",
    version.ref = "junit-jupiter-engine" }

[plugins]
kotlin-jvm = { id = "org.jetbrains.kotlin.jvm", version = "2.1.0" }
```

- Accédez au répertoire `app` et ouvrez le fichier `build.gradle.kts`. Remplacez son contenu par le code suivant, puis enregistrez vos modifications.

```
plugins {
    alias(libs.plugins.kotlin.jvm)
    application
}

dependencies {
    implementation(awssdk.services.s3) // Add dependency on the Kit AWS SDK pour
    Kotlin's S3 client.
```

```
testImplementation("org.jetbrains.kotlin:kotlin-test-junit5")
testImplementation(libs.junit.jupiter.engine)
testRuntimeOnly("org.junit.platform:junit-platform-launcher")
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

application {
    mainClass = "org.example.AppKt"
}

tasks.named<Test>("test") {
    useJUnitPlatform()
}
```

La `dependencies` section contient une `implementation` entrée pour le module Amazon S3 du Kit AWS SDK pour Kotlin. Le compilateur Gradle est configuré pour utiliser Java 21 dans la `java` section.

Étape 3 : Écrire le code

Une fois le projet créé et configuré, modifiez la classe par défaut du projet App pour utiliser l'exemple de code suivant.

1. Dans le dossier de votre projet `app`, naviguez jusqu'au répertoire `src/main/kotlin/org/example`. Ouvrez le fichier `App.kt`.
2. Remplacez son contenu par le code suivant et enregistrez le fichier.

```
package org.example

import aws.sdk.kotlin.services.s3.*
import aws.sdk.kotlin.services.s3.model.BucketLocationConstraint
import aws.smithy.kotlin.runtime.content.ByteString
import kotlinx.coroutines.runBlocking
import java.util.UUID
```

```
val REGION = "us-west-2"
val BUCKET = "bucket-#{UUID.randomUUID()}"
val KEY = "key"

fun main(): Unit = runBlocking {
    S3Client
        .fromEnvironment { region = REGION }
        .use { s3 ->
            setupTutorial(s3)

            println("Creating object $BUCKET/$KEY...")

            s3.putObject {
                bucket = BUCKET
                key = KEY
                body = ByteString.fromString("Testing with the Kotlin SDK")
            }

            println("Object $BUCKET/$KEY created successfully!")

            cleanUp(s3)
        }
}

suspend fun setupTutorial(s3: S3Client) {
    println("Creating bucket $BUCKET...")
    s3.createBucket {
        bucket = BUCKET
        if (REGION != "us-east-1") { // Do not set location constraint for us-east-1.
            createBucketConfiguration {
                locationConstraint = BucketLocationConstraint.fromValue(REGION)
            }
        }
    }
    println("Bucket $BUCKET created successfully!")
}

suspend fun cleanUp(s3: S3Client) {
    println("Deleting object $BUCKET/$KEY...")
    s3.deleteObject {
        bucket = BUCKET
        key = KEY
    }
    println("Object $BUCKET/$KEY deleted successfully!")
}
```

```
println("Deleting bucket $BUCKET...")
s3.deleteBucket {
    bucket = BUCKET
}
println("Bucket $BUCKET deleted successfully!")
}
```

Étape 4 : créer et exécuter l'application

Une fois le projet créé et contenant l'exemple de classe, créez et exécutez l'application.

1. Ouvrez un terminal ou une fenêtre d'invite de commande et accédez au répertoire de votre projet `getstarted`.
2. Utilisez la commande suivante pour créer et exécuter votre application :

```
gradle run
```

Note

Si vous obtenez un `IdentityProviderException`, il se peut que vous n'ayez pas de session d'authentification unique active. Exécutez la commande `aws sso login` AWS CLI pour démarrer une nouvelle session.

L'application appelle l'opération d'API [CreateBucket](#) pour créer un nouveau compartiment S3, puis appelle [PutObject](#) pour placer un nouvel objet dans le nouveau compartiment S3.

Dans la `cleanup()` fonction située à la fin, l'application supprime l'objet puis le compartiment S3.

Pour voir les résultats dans la console Amazon S3

1. Dans `App.kt`, commentez la ligne `cleanup(s3)` de la `runBlocking` section et enregistrez le fichier.
2. Reconstituez le projet et placez un nouvel objet dans un nouveau compartiment S3 en exécutant `gradle run`.

3. Connectez-vous à la [console Amazon S3](#) pour afficher le nouvel objet dans le nouveau compartiment S3.

Après avoir visualisé l'objet, supprimez le compartiment S3.

Réussite

Si votre projet Gradle a été créé et exécuté sans erreur, alors félicitations. Vous avez créé avec succès votre première application Kotlin à l'aide du Kit AWS SDK pour Kotlin.

Nettoyage

Lorsque vous avez terminé de développer votre nouvelle application, supprimez toutes les AWS ressources que vous avez créées au cours de ce didacticiel pour éviter d'encourir des frais. Vous pouvez également supprimer ou archiver le dossier de projet (get-started) que vous avez créé à l'étape 2.

Pour nettoyer les ressources, procédez comme suit :

- Si vous avez commenté l'appel à la `cleanup()` fonction, supprimez le compartiment S3 à l'aide de la [console Amazon S3](#).

Étapes suivantes

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

- [Étapes de configuration supplémentaires pour travailler avec le SDK pour Kotlin](#)
- [Configuration du SDK pour Kotlin](#)
- [Utilisation du SDK pour Kotlin](#)
- [Sécurité du SDK pour Kotlin](#)

Configurez le Kit AWS SDK pour Kotlin

Pour demander à Services AWS utiliser le Kit AWS SDK pour Kotlin, vous avez besoin des éléments suivants :

- Possibilité de se connecter au portail d' AWS accès
- Autorisation d'utiliser les AWS ressources dont votre application a besoin
- 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'identification IAM Identity Center afin que le SDK puisse obtenir des informations d'identification. AWS
 - Le `credentials` fichier contient des informations d'identification temporaires
 - [Un outil d'automatisation du build tel que Gradle ou Maven](#)
- Une session de portail AWS d'accès active lorsque vous êtes prêt à exécuter votre application

Dans cette rubrique

- [Configuration de base](#)
- [Création de fichiers de construction de projets](#)
- [Codez votre projet Kotlin à l'aide du SDK pour Kotlin](#)

Configuration de base

Présentation

Pour développer avec succès des applications qui accèdent à l' Services AWS aide de Kit AWS SDK pour Kotlin, les conditions suivantes doivent être remplies.

- Vous devez être en mesure de vous [connecter au portail AWS d'accès](#) disponible dans le AWS IAM Identity Center.
- Les [autorisations du rôle IAM](#) configuré pour le SDK doivent autoriser l'accès à Services AWS ce dont votre application a besoin. Les autorisations associées à la politique `PowerUserAccess` AWS gérée sont suffisantes pour répondre à 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 qu'IntelliJ IDEA ou Eclipse.](#)

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.

- 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 organisation 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 pour Kotlin. Suivez ces instructions pour ajouter des informations d'identification temporaires au `~/.aws/credentials` fichier.

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

Possibilité de connexion au portail d' AWS accès

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

Si vous ne connaissez pas le portail d' AWS accès, suivez les instructions relatives à l'accès au compte figurant dans la rubrique [Authentification du centre d'identité IAM](#) du guide de référence sur les outils AWS SDKs et.

Configurer l'accès à authentification unique pour le SDK

Une fois que vous avez terminé l'étape 2 de la [section Accès par programmation](#) afin que le SDK utilise 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 pour Kotlin utilise la configuration du fournisseur de jetons SSO du profil pour acquérir des informations d'identification avant d'envoyer des demandes à AWS. La valeur `sso_role_name`, qui est un rôle IAM connecté à un ensemble d'autorisations IAM Identity Center, doit autoriser l'accès aux Services AWS utilisés dans 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 des outils AWS SDKs et.

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

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
```

Comme votre profil est configuré par défaut, il n'est pas nécessaire d'appeler la commande avec une `--profile` option. 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 commande AWS CLI 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 de portail d'accès AWS active et que vous 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 pour Kotlin.

Installation de Java et d'un outil de compilation

Votre environnement de développement a besoin des éléments suivants :

- JDK 8 ou version ultérieure. [Il Kit AWS SDK pour Kotlin fonctionne avec le kit de développement Oracle Java SE et avec les distributions du kit de développement Open Java \(OpenJDK\) Amazon Correttotelles que Red Hat OpenJDK et JDK. AdoptOpen](#)
- 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 <http://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/>

Utiliser des informations d'identification temporaires

Au lieu de [configurer l'accès à authentification unique IAM Identity Center](#) pour le SDK, vous pouvez configurer votre environnement de développement à l'aide d'informations d'identification temporaires.

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

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

```
[default]
aws_access_key_id=<value from AWS access portal>
aws_secret_access_key=<value from AWS access portal>
aws_session_token=<value from AWS access portal>
```

3. Enregistrez le fichier. Le fichier `~/.aws/credentials` devrait maintenant exister sur votre système de développement local. Ce fichier contient le [profil \[par défaut\]](#) que le SDK pour Kotlin utilise si aucun profil nommé spécifique n'est spécifié.
4. [Connectez-vous au portail d' AWS accès](#)

Gradle

Il Kit AWS SDK pour Kotlin publie un [catalogue de versions et une nomenclature \(BOM\) de Gradle](#) qui peuvent vous aider à découvrir les noms des dépendances et à synchroniser les numéros de version entre plusieurs artefacts.

Notez que les catalogues de versions sont une fonctionnalité d'aperçu de Gradle avant la version 8. Selon la version de Gradle que vous utilisez, vous devrez peut-être vous inscrire via l'API [Feature Preview](#).

Pour utiliser un catalogue de versions Gradle

1. Dans votre `settings.gradle.kts` fichier, ajoutez un `versionCatalogs` bloc à l'intérieur du `dependencyResolutionManagement` bloc.

Le fichier d'exemple suivant configure le catalogue de Kit AWS SDK pour Kotlin versions. Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.

```
plugins {
    id("org.gradle.toolchains.foojay-resolver-convention") version "X.Y.Z"
}
rootProject.name = "your-project-name"

dependencyResolutionManagement {
    repositories {
        mavenCentral()
    }

    versionCatalogs {
        create("awssdk") {
            from("aws.sdk.kotlin:version-catalog:X.Y.Z")
        }
    }
}
```

2. Déclarez les dépendances en `build.gradle.kts` utilisant les identifiants sécurisés mis à disposition par le catalogue de versions.

Le fichier d'exemple suivant déclare les dépendances pour sept Services AWS.

```
plugins {
    kotlin("jvm") version "X.Y.Z"
```



```
    application
  }

group = "org.example"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    implementation(platform(awssdk.bom))
    implementation(platform("org.apache.logging.log4j:log4j-bom:X.Y.Z"))

    implementation(awssdk.services.s3)
    implementation(awssdk.services.dynamodb)
    implementation(awssdk.services.iam)
    implementation(awssdk.services.cloudwatch)
    implementation(awssdk.services.cognitoidentityprovider)
    implementation(awssdk.services.sns)
    implementation(awssdk.services.pinpoint)
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")

    // Test dependency.
    testImplementation(kotlin("test"))
}

tasks.test {
    useJUnitPlatform()
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(X*)
    }
}

application {
    mainClass = "org.example.AppKt"
}
```

* Version Java, par exemple 17 ou 21.

Maven

Le pom.xml fichier d'exemple suivant comporte des dépendances pour sept Services AWS. Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.

```
<?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/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>setup</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <aws.sdk.kotlin.version>X.Y.Z</aws.sdk.kotlin.version>
        <kotlin.version>X.Y.Z</kotlin.version>
        <log4j.version>X.Y.Z</log4j.version>
        <junit.jupiter.version>X.Y.Z</junit.jupiter.version>
        <jvm.version>X*</jvm.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>aws.sdk.kotlin</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.sdk.kotlin.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>org.apache.logging.log4j</groupId>
                <artifactId>log4j-bom</artifactId>
                <version>${log4j.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

```
<dependencies>
  <dependency>
    <groupId>aws.sdk.kotlin</groupId>
    <artifactId>s3-jvm</artifactId>
  </dependency>
  <dependency>
    <groupId>aws.sdk.kotlin</groupId>
    <artifactId>dynamodb-jvm</artifactId>
  </dependency>
  <dependency>
    <groupId>aws.sdk.kotlin</groupId>
    <artifactId>iam-jvm</artifactId>
  </dependency>
  <dependency>
    <groupId>aws.sdk.kotlin</groupId>
    <artifactId>cloudwatch-jvm</artifactId>
  </dependency>
  <dependency>
    <groupId>aws.sdk.kotlin</groupId>
    <artifactId>cognitoidentityprovider-jvm</artifactId>
  </dependency>
  <dependency>
    <groupId>aws.sdk.kotlin</groupId>
    <artifactId>sns-jvm</artifactId>
  </dependency>
  <dependency>
    <groupId>aws.sdk.kotlin</groupId>
    <artifactId>pinpoint-jvm</artifactId>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j2-impl</artifactId>
  </dependency>

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

```
        <artifactId>junit-jupiter</artifactId>
        <version>${junit.jupiter.version}</version>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <sourceDirectory>src/main/kotlin</sourceDirectory>
    <testSourceDirectory>src/test/kotlin</testSourceDirectory>

    <plugins>
        <plugin>
            <groupId>org.jetbrains.kotlin</groupId>
            <artifactId>kotlin-maven-plugin</artifactId>
            <version>${kotlin.version}</version>
            <executions>
                <execution>
                    <id>compile</id>
                    <phase>compile</phase>
                    <goals>
                        <goal>compile</goal>
                    </goals>
                </execution>
                <execution>
                    <id>test-compile</id>
                    <phase>test-compile</phase>
                    <goals>
                        <goal>test-compile</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <jvmTarget>${jvm.version}</jvmTarget>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>
```

* Version Java, par exemple 17 ou 21.

Codez votre projet Kotlin à l'aide du SDK pour Kotlin

Maintenant, le plaisir commence. Au fur et à mesure que vous développez votre application, vous pouvez vous référer à la [référence des Kit AWS SDK pour Kotlin API](#) pour obtenir des informations complètes sur les opérations de l'API. Utilisez les liens suivants pour obtenir des informations générales sur l'API Kotlin :

- [Référence d'API de bibliothèque standard](#)
- [Présentation des Coroutines](#)
- [API Coroutines](#)

Connectez-vous à l'aide du AWS CLI

Chaque fois que vous exécutez un programme qui y accède Services AWS, vous avez besoin d'une session de portail AWS d'accès active. Pour ce faire, exécutez la commande d' .

```
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 votre configuration d'authentification unique IAM Identity Center 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`.

Configurez le Kit AWS SDK pour Kotlin

Cette section explique comment configurer un client de service à l'aide du Kit AWS SDK pour Kotlin. Pour plus d'informations, consultez le [Guide de référence du SDK et des outils](#), qui inclut une présentation de la configuration applicable à tous AWS SDKs.

Table des matières

- [Création d'un client de service](#)
 - [Configurer un client dans le code](#)
 - [Configuration d'un client depuis l'environnement](#)
 - [Fermez le client](#)
- [Région AWS sélection](#)
 - [Chaîne de fournisseurs de régions par défaut](#)
- [Fournisseurs d'informations d'identification](#)
 - [La chaîne de fournisseurs d'informations d'identification par défaut](#)
 - [En savoir plus sur la chaîne de fournisseurs d'informations d'identification par défaut](#)
 - [Fournisseur d'informations d'identification explicite](#)
- [Configuration des points de terminaison clients](#)
 - [Configuration personnalisée](#)
 - [Définir endpointUrl](#)
 - [Définir endpointProvider](#)
 - [Propriétés de EndpointProvider](#)
 - [endpointUrl ou endpointProvider](#)
 - [Remarque à propos d'Amazon S3](#)
 - [Exemples](#)
 - [Exemple de endpointUrl](#)
 - [Exemple de endpointProvider](#)
 - [endpointUrl et endpointProvider](#)
- [HTTP](#)
 - [Configuration du client HTTP](#)
 - [Configuration de base](#)

- [Importations](#)
- [Code](#)
- [Spécifier un type de moteur HTTP](#)
 - [Importations](#)
 - [Code](#)
 - [Utilisation de la OkHttp4Engine](#)
 - [Utiliser un client HTTP explicite](#)
 - [Importations](#)
 - [Code](#)
- [Utiliser un proxy HTTP](#)
 - [Utiliser les propriétés du système JVM](#)
 - [Utiliser des variables d'environnement](#)
 - [Utiliser un proxy sur les EC2 instances](#)
- [Intercepteurs HTTP](#)
 - [Enregistrement de l'intercepteur](#)
 - [Intercepteur pour toutes les opérations des clients de service](#)
 - [Intercepteur pour des opérations spécifiques uniquement](#)
- [Appliquer une version minimale de TLS](#)
 - [Configuration du moteur HTTP](#)
 - [Définissez la propriété du système sdk.minTls JVM](#)
 - [Définissez la variable d'SDK_MIN_TLSenvironnement](#)
- [Nouvelle tentative](#)
 - [Configuration de nouvelle tentative par défaut](#)
 - [Nombre maximum de tentatives](#)
 - [Retards et retards](#)
 - [Réessayez le bucket de jetons](#)
 - [Tentatives adaptatives](#)
- [Observabilité](#)
 - [Configurez un TelemetryProvider](#)
 - [Configuration du fournisseur de télémétrie global par défaut](#)

- [Configuration d'un fournisseur de télémétrie pour un client de service spécifique](#)
- [Métriques](#)
- [Journalisation](#)
 - [Spécifier le mode journal pour les messages au niveau du fil](#)
 - [Définir le mode journal dans le code](#)
 - [Définir le mode journal depuis l'environnement](#)
- [Fournisseurs de services de télémétrie](#)
 - [Configuration du fournisseur OpenTelemetry de télémétrie basé](#)
 - [Prérequis](#)
 - [Configuration du kit SDK](#)
 - [Ressources](#)
- [Remplacer la configuration du client de service](#)
 - [Cycle de vie d'un client remplacé](#)
 - [Ressources partagées entre les clients](#)

Création d'un client de service

Pour faire une demande à un Service AWS, vous devez d'abord instancier un client pour ce service.

Vous pouvez configurer des paramètres courants pour les clients de service, tels que le client HTTP à utiliser, le niveau de journalisation et la configuration des nouvelles tentatives. En outre, chaque client de service a besoin d'un fournisseur d'informations d'identification Région AWS et d'un fournisseur d'informations d'identification. Le SDK utilise ces valeurs pour envoyer des demandes à la région appropriée et pour signer les demandes avec les informations d'identification correctes.

Vous pouvez spécifier ces valeurs par programmation dans le code ou les charger automatiquement depuis l'environnement.

Configurer un client dans le code

Pour configurer un client de service avec des valeurs spécifiques, vous pouvez les spécifier dans une fonction lambda transmise à la méthode d'usine du client de service, comme indiqué dans l'extrait suivant.

```
val dynamoDbClient = DynamoDbClient {
```



```
region = "us-east-1"
credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
}
```

Toutes les valeurs que vous ne spécifiez pas dans le bloc de configuration sont définies par défaut. Par exemple, si vous ne spécifiez pas de fournisseur d'informations d'identification comme le faisait le code précédent, le fournisseur d'informations d'identification utilise [par défaut la chaîne de fournisseurs d'informations d'identification par défaut](#).

Warning

Certaines propriétés, par exemple, `region` n'ont pas de valeur par défaut. Vous devez les spécifier explicitement dans le bloc de configuration lorsque vous utilisez la configuration programmatique. Si le SDK ne parvient pas à résoudre la propriété, les demandes d'API risquent d'échouer.

Configuration d'un client depuis l'environnement

Lors de la création d'un client de service, le SDK peut inspecter des emplacements dans l'environnement d'exécution actuel afin de déterminer certaines propriétés de configuration. Ces emplacements incluent les [fichiers de configuration et d'identification partagés](#), les [variables d'environnement](#) et les [propriétés du système JVM](#). Les propriétés pouvant être résolues incluent [AWS la région](#), [la stratégie de nouvelle tentative](#), [le mode journal](#), etc. Pour plus d'informations sur tous les paramètres que le SDK peut résoudre à partir de l'environnement d'exécution, consultez le [Guide de référence des paramètres AWS SDKs et des outils](#).

Pour créer un client avec une configuration basée sur l'environnement, utilisez la méthode statique `suspend fun fromEnvironment()` sur l'interface du client de service :

```
val dynamoDbClient = DynamoDbClient.fromEnvironment()
```

La création d'un client de cette manière est utile lors de l'exécution sur Amazon EC2 ou dans tout autre contexte dans lequel la configuration d'un client de service est disponible depuis l'environnement. AWS Lambda Cela dissocie votre code de l'environnement dans lequel il s'exécute et facilite le déploiement de votre application dans plusieurs régions sans modifier le code.

En outre, vous pouvez remplacer des propriétés spécifiques en transmettant un bloc lambda à `fromEnvironment`. L'exemple suivant charge certaines propriétés de configuration depuis

l'environnement (par exemple, Region) mais remplace spécifiquement le fournisseur d'informations d'identification pour utiliser les informations d'identification d'un profil.

```
val dynamoDbClient = DynamoDbClient.fromEnvironment {  
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")  
}
```

Le SDK utilise des valeurs par défaut pour toutes les propriétés de configuration qui ne peuvent pas être déterminées à partir des paramètres de programmation ou de l'environnement. Par exemple, si vous ne spécifiez aucun fournisseur d'informations d'identification dans le code ou dans un paramètre d'environnement, le fournisseur d'informations d'identification utilise [par défaut la chaîne de fournisseurs d'informations d'identification par défaut](#).

Warning

Certaines propriétés, telles que Region, n'ont pas de valeur par défaut. Vous devez les spécifier dans un paramètre d'environnement ou explicitement dans le bloc de configuration. Si le SDK ne parvient pas à résoudre la propriété, les demandes d'API risquent d'échouer.

Note

Bien que les propriétés liées aux informations d'identification, telles que les clés d'accès temporaires et la configuration SSO, soient présentes dans l'environnement d'exécution, les valeurs ne proviennent pas du client au moment de la création. Au lieu de cela, les valeurs sont accessibles par la couche fournisseur d'informations d'identification à chaque demande.

Fermez le client

Lorsque vous n'avez plus besoin du client de service, fermez-le pour libérer les ressources qu'il utilise :

```
val dynamoDbClient = DynamoDbClient.fromEnvironment()  
// Invoke several DynamoDB operations.  
dynamoDbClient.close()
```

Comme les clients de service étendent l'[Closeable](#) interface, vous pouvez utiliser l'[use](#) extension pour fermer automatiquement le client à la fin d'un bloc, comme indiqué dans l'extrait suivant.

```
DynamoDbClient.fromEnvironment().use { dynamoDbClient ->
    // Invoke several DynamoDB operations.
}
```

Dans l'exemple précédent, le bloc lambda reçoit une référence au client qui vient d'être créé. Vous pouvez appeler des opérations sur cette référence client et lorsque le blocage est terminé, notamment en lançant une exception, le client est fermé.

Région AWS sélection

Avec Régions AWS, vous pouvez accéder à Services AWS ceux qui opèrent dans une zone géographique spécifique. Cela peut être utile pour la redondance et pour maintenir vos données et vos applications en cours d'exécution à proximité de l'endroit où vous-même et vos utilisateurs y accédez.

Chaîne de fournisseurs de régions par défaut

Lors du chargement de la configuration d'un client de service [depuis l'environnement](#), le processus de recherche suivant est utilisé :

1. Toute région explicite définie sur le générateur.
2. La propriété du système `aws.region` JVM est vérifiée. Si elle est définie, cette région est utilisée dans la configuration du client.
3. La variable d'environnement `AWS_REGION` est contrôlée. Si elle est définie, cette région est utilisée dans la configuration du client.
 - a. Remarque : Cette variable d'environnement est définie par le conteneur Lambda.
4. Le SDK vérifie le fichier de configuration AWS partagé. Si la `region` propriété est définie pour le profil actif, le SDK l'utilise.
 - a. La variable d'environnement `AWS_CONFIG_FILE` peut être utilisée pour personnaliser l'emplacement du fichier de configuration partagé.
 - b. La propriété du système `aws.profile` JVM ou la variable d'`AWS_PROFILE` environnement peuvent être utilisées pour personnaliser le profil chargé par le SDK.
5. Le SDK tente d'utiliser le service de métadonnées d' EC2 instance Amazon pour déterminer la région de l' EC2 instance en cours d'exécution.
6. Si la région n'est toujours pas résolue à ce stade, la création du client échoue à une exception près.

Fournisseurs d'informations d'identification

⚠ L'ordre dans lequel la chaîne de fournisseurs d'informations d'identification par défaut résout les informations d'identification a changé avec la version 1.4.0. Pour plus de détails, consultez la note ci-dessous.

Pour envoyer des demandes à Amazon Web Services à l'aide du Kit AWS SDK pour Kotlin, le SDK utilise des informations d'identification signées cryptographiquement émises par AWS. Pour obtenir les informations d'identification, le SDK peut utiliser des paramètres de configuration situés à plusieurs endroits, par exemple les propriétés du système JVM, les variables d'environnement, `credentials` les fichiers partagés AWS `config` et les métadonnées des EC2 instances Amazon.

Le SDK utilise l'abstraction du fournisseur d'informations d'identification pour simplifier le processus de récupération des informations d'identification à partir de diverses sources. Le SDK contient [plusieurs implémentations de fournisseurs d'informations d'identification](#).

Par exemple, si la configuration récupérée inclut les paramètres d'accès à authentification unique IAM Identity Center à partir du `config` fichier partagé, le SDK travaille avec le IAM Identity Center pour récupérer les informations d'identification temporaires qu'il utilise pour effectuer une demande. Services AWS Dans le cadre de cette approche d'acquisition d'informations d'identification, le SDK utilise le fournisseur IAM Identity Center (également connu sous le nom de fournisseur d'informations d'identification SSO). La [section de configuration](#) de ce guide décrit cette configuration.

Pour utiliser un fournisseur d'informations d'identification spécifique, vous pouvez en spécifier un lorsque vous créez un client de service. Vous pouvez également utiliser la chaîne de fournisseurs d'informations d'identification par défaut pour rechercher automatiquement les paramètres de configuration.

La chaîne de fournisseurs d'informations d'identification par défaut

Lorsqu'il n'est pas explicitement spécifié lors de la construction du client, le SDK pour Kotlin utilise un fournisseur d'informations d'identification qui vérifie séquentiellement chaque endroit où vous pouvez fournir des informations d'identification. Ce fournisseur d'informations d'identification par défaut est implémenté sous la forme d'une chaîne de fournisseurs d'informations d'identification.

Pour utiliser la chaîne par défaut pour fournir des informations d'identification dans votre application, créez un client de service sans fournir explicitement de `credentialsProvider` propriété.

```
val ddb = DynamoDbClient {  
    region = "us-east-2"  
}
```

Pour plus d'informations sur la création d'un client de service, consultez [la section Création et configuration d'un client](#).

En savoir plus sur la chaîne de fournisseurs d'informations d'identification par défaut

La chaîne de fournisseurs d'informations d'identification par défaut recherche la configuration des informations d'identification à l'aide de la séquence prédéfinie suivante. Lorsque les paramètres configurés fournissent des informations d'identification valides, la chaîne s'arrête.

1. [AWS clés d'accès \(propriétés du système JVM\)](#)

Le SDK recherche les propriétés `aws.accessKeyId` et `aws.secretAccessKey`, et du système `aws.sessionToken` JVM.

2. [AWS clés d'accès \(variables d'environnement\)](#)

Le SDK tente de charger les informations d'identification à partir `AWS_ACCESS_KEY_ID` des variables `AWS_SECRET_ACCESS_KEY` d'`AWS_SESSION_TOKEN` environnement et.

3. [Jeton d'identité Web](#)

Le SDK recherche les variables `AWS_WEB_IDENTITY_TOKEN_FILE` d'environnement et/ ou `AWS_ROLE_ARN` (ou les propriétés du système JVM `aws.webIdentityTokenFile` et `aws.roleArn`). Sur la base des informations du jeton et du rôle, le SDK acquiert des informations d'identification temporaires.

4. [Un profil dans un fichier de configuration](#)

Dans cette étape, le SDK utilise les paramètres associés à un profil. Par défaut, le SDK utilise le partage `AWS config` et `credentials` les fichiers, mais si la variable d'`AWS_CONFIG_FILE` environnement est définie, le SDK utilise cette valeur. Si la variable d'`AWS_PROFILE` environnement (ou la propriété du système `aws.profile` JVM) n'est pas définie, le SDK recherche le profil « par défaut », sinon il recherche le profil correspondant `AWS_PROFILE` 's à la valeur.

Le SDK recherche le profil en fonction de la configuration décrite dans le paragraphe précédent et utilise les paramètres qui y sont définis. Si les paramètres trouvés par le SDK contiennent une

combinaison de paramètres correspondant à différentes approches de fourniture d'informations d'identification, le SDK utilise l'ordre suivant :

- a. [AWS clés d'accès \(fichier de configuration\)](#) - Le SDK utilise les paramètres pour `aws_access_key_id`, `aws_secret_access_key`, `aws_session_token`.
- b. [Assumer la configuration des rôles](#) : si le SDK trouve `role_arn` `source_profile` et/ou des `credential_source` paramètres, il tente d'assumer un rôle. Si le SDK trouve le `source_profile` paramètre, il utilise les informations d'identification d'un autre profil pour recevoir des informations d'identification temporaires pour le rôle spécifié par `role_arn`. Si le SDK trouve le `credential_source` paramètre, il extrait les informations d'identification d'un conteneur Amazon ECS, d'une EC2 instance Amazon ou de variables d'environnement en fonction de la valeur du `credential_source` paramètre. Il utilise ensuite ces informations d'identification pour acquérir des informations d'identification temporaires pour le rôle.

Un profil doit contenir le `source_profile` paramètre ou le `credential_source` paramètre, mais pas les deux.

- c. [Configuration du jeton d'identité Web](#) : si le SDK trouve `role_arn` et définit `web_identity_token_file` les paramètres, il acquiert des informations d'identification temporaires pour accéder aux AWS ressources en fonction du jeton `role_arn` et du jeton.
- d. [Configuration du jeton SSO](#) : si le SDK trouve `sso_session` des `sso_role_name` paramètres (ainsi qu'une `sso-session` section associée dans les fichiers de configuration), le SDK récupère les informations d'identification temporaires auprès du service IAM Identity Center. `sso_account_id`
- e. [Configuration SSO traditionnelle](#) : si le SDK trouve `sso_start_url`, et définit les `sso_role_name` paramètres `sso_region` `sso_account_id`, il récupère les informations d'identification temporaires auprès du service IAM Identity Center.
- f. [Configuration du processus](#) : si le SDK trouve un `credential_process` paramètre, il utilise la valeur du chemin pour appeler un processus et obtenir des informations d'identification temporaires.

5. [Informations d'identification du conteneur](#)

Le SDK recherche les variables d'environnement

`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` ou

ou `AWS_CONTAINER_CREDENTIALS_FULL_URI`.

`AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE` `AWS_CONTAINER_AUTHORIZATION_TOKEN`

Il utilise ces valeurs pour charger les informations d'identification depuis le point de terminaison HTTP spécifié via une requête GET.

6. Informations d'identification IMDS

Le SDK tente de récupérer les informations d'identification auprès du [service de métadonnées d'instance](#) sur le point de terminaison HTTP par défaut ou configuré. Le SDK prend uniquement en charge [IMDSv2](#).

Si les informations d'identification ne sont toujours pas résolues à ce stade, la création du client échoue avec une exception.

Remarque : Modification de l'ordre de résolution des informations d'identification

L'ordre de résolution des informations d'identification décrit ci-dessus est en vigueur pour la 1.4.x+ sortie du SDK pour Kotlin. Avant la 1.4.0 sortie, les articles numéros 3 et 4 ont été échangés et l'élément 4a actuel a suivi l'élément 4f actuel.

Fournisseur d'informations d'identification explicite

Au lieu d'utiliser la chaîne de fournisseurs par défaut, vous pouvez spécifier un fournisseur d'informations d'identification spécifique ou une chaîne personnalisée (`CredentialsProviderChain`) que le SDK doit utiliser. Par exemple, si vous définissez les informations d'identification par défaut à l'aide de variables d'environnement, fournissez un `EnvironmentCredentialsProvider` au générateur de clients, comme dans l'extrait de code suivant.

```
val ddb = DynamoDbClient {
    region = "us-east-1"
    credentialsProvider = EnvironmentCredentialsProvider()
}
```

Note

La chaîne par défaut met en cache les informations d'identification, mais pas les fournisseurs autonomes. Vous pouvez encapsuler n'importe quel fournisseur d'informations d'identification à l'aide `CachedCredentialsProvider` de la classe pour éviter de récupérer inutilement des informations d'identification à chaque appel d'API. Le fournisseur mis en cache ne

recupère les nouvelles informations d'identification que lorsque les informations actuelles expirent.

Note

Vous pouvez implémenter votre propre fournisseur d'informations d'identification ou votre propre chaîne de fournisseurs en implémentant l'`CredentialsProvider` interface.

Configuration des points de terminaison clients

Lorsqu'il Kit AWS SDK pour Kotlin appelle un Service AWS, l'une de ses premières étapes consiste à déterminer où acheminer la demande. Ce processus est connu sous le nom de résolution des terminaux.

Vous pouvez configurer la résolution des points de terminaison pour le SDK lorsque vous créez un client de service. La configuration par défaut pour la résolution des points de terminaison est généralement correcte, mais plusieurs raisons peuvent vous amener à modifier la configuration par défaut. Voici deux exemples de raisons :

- Envoyez des demandes à une version préliminaire d'un service ou à un déploiement local d'un service.
- Accès à des fonctionnalités de service spécifiques non encore modélisées dans le SDK.

Warning

La résolution des points de terminaison est une rubrique avancée du SDK. Si vous modifiez les paramètres par défaut, vous risquez de casser votre code. Les paramètres par défaut doivent s'appliquer à la plupart des utilisateurs dans les environnements de production.

Configuration personnalisée

Vous pouvez personnaliser la résolution des points de terminaison d'un client de service à l'aide de deux propriétés disponibles lorsque vous créez le client :

1. `endpointUrl`: `Url`
2. `endpointProvider`: `EndpointProvider`

Définir `endpointUrl`

Vous pouvez définir une valeur pour `endpointUrl` pour indiquer un nom d'hôte « de base » pour le service. Cette valeur n'est toutefois pas définitive puisqu'elle est transmise en tant que paramètre à l'`EndpointProvider` instance du client. L'`EndpointProvider` implémentation peut ensuite inspecter et éventuellement modifier cette valeur pour déterminer le point final.

Par exemple, si vous spécifiez une `endpointUrl` valeur pour un client Amazon Simple Storage Service (Amazon S3) et que vous effectuez `GetObject` une opération, l'implémentation du fournisseur de point de terminaison par défaut injecte le nom du compartiment dans la valeur du nom d'hôte.

Dans la pratique, les utilisateurs définissent une `endpointUrl` valeur qui pointe vers une instance de développement ou de prévisualisation d'un service.

Définir `endpointProvider`

La `EndpointProvider` mise en œuvre d'un client de service détermine la résolution finale du point final. L'`EndpointProvider` interface illustrée dans le bloc de code suivant expose la `resolveEndpoint` méthode.

```
public fun interface EndpointProvider<T> {  
    public suspend fun resolveEndpoint(params: T): Endpoint  
}
```

Un client de service appelle la `resolveEndpoint` méthode pour chaque demande. Le client du service utilise la `Endpoint` valeur renvoyée par le fournisseur sans autre modification.

Propriétés de `EndpointProvider`

La `resolveEndpoint` méthode accepte un `EndpointParameters` objet spécifique au service qui contient les propriétés utilisées dans la résolution des points de terminaison.

Chaque service inclut les propriétés de base suivantes.

Nom	Type	Description
region	Chaîne	La AWS région du client
endpoint	Chaîne	Une représentation sous forme de chaîne de l'ensemble de valeurs de <code>endpointUrl</code>
useFips	Booléen	Si les points de terminaison FIPS sont activés dans la configuration du client
useDualStack	Booléen	Si les points de terminaison à double pile sont activés dans la configuration du client

Les services peuvent spécifier des propriétés supplémentaires requises pour la résolution. Par exemple, Amazon S3 [S3EndpointParameters](#) inclut le nom du compartiment ainsi que plusieurs paramètres de fonctionnalités spécifiques à Amazon S3. Par exemple, la `forcePathStyle` propriété détermine si l'adressage d'hôte virtuel peut être utilisé.

Si vous implémentez votre propre fournisseur, vous ne devriez pas avoir à créer votre propre instance de `EndpointParameters`. Le SDK fournit les propriétés de chaque demande et les transmet à votre implémentation de `resolveEndpoint`.

endpointUrl ou **endpointProvider**

Il est important de comprendre que les deux instructions suivantes NE produisent PAS pour les clients un comportement de résolution de point de terminaison équivalent :

```
// Use endpointUrl.
S3Client.fromEnvironment {
    endpointUrl = Url.parse("https://endpoint.example")
}

// Use endpointProvider.
S3Client.fromEnvironment {
    endpointProvider = object : S3EndpointProvider {
```

```
        override fun resolveEndpoint(params: S3EndpointParameters): Endpoint =
            Endpoint("https://endpoint.example")
    }
}
```

L'instruction qui définit la `endpointUrl` propriété spécifie une URL de base transmise au fournisseur (par défaut), qui peut être modifiée dans le cadre de la résolution du point de terminaison.

L'instruction qui définit le `endpointProvider` spécifie l'URL finale qu'il `S3Client` utilise.

Bien que vous puissiez définir les deux propriétés, dans la plupart des cas nécessitant une personnalisation, vous fournissez l'une d'entre elles. En tant qu'utilisateur général du SDK, vous fournissez le plus souvent une `endpointUrl` valeur.

Remarque à propos d'Amazon S3

Amazon S3 est un service complexe dont bon nombre de fonctionnalités sont modélisées par le biais de personnalisations personnalisées des points de terminaison, telles que l'hébergement virtuel de compartiments. L'hébergement virtuel est une fonctionnalité d'Amazon S3 dans laquelle le nom du compartiment est inséré dans le nom d'hôte.

C'est pourquoi nous vous recommandons de ne pas remplacer l'`EndpointProvider` implémentation dans un client de service Amazon S3. Si vous devez étendre son comportement de résolution, par exemple en envoyant des requêtes à une pile de développement locale en tenant compte des points de terminaison supplémentaires, nous vous recommandons d'encapsuler l'implémentation par défaut. L'`EndpointProvider` exemple suivant montre un exemple de mise en œuvre de cette approche.

Exemples

Exemple de `endpointUrl`

L'extrait de code suivant montre comment le point de terminaison du service général peut être remplacé pour un client Amazon S3.

```
val client = S3Client.fromEnvironment {
    endpointUrl = Url.parse("https://custom-s3-endpoint.local")
    // EndpointProvider is left as the default.
}
```

Exemple de `endpointProvider`

L'extrait de code suivant montre comment fournir un fournisseur de point de terminaison personnalisé qui intègre l'implémentation par défaut pour Amazon S3.

```
import aws.sdk.kotlin.services.s3.endpoints.DefaultS3EndpointProvider
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointParameters
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointProvider
import aws.smithy.kotlin.runtime.client.endpoints.Endpoint

public class CustomS3EndpointProvider : S3EndpointProvider {
    override suspend fun resolveEndpoint(params: S3EndpointParameters) =
        if (/* Input params indicate we must route another endpoint for whatever
reason. */) {
            Endpoint(/* ... */)
        } else {
            // Fall back to the default resolution.
            DefaultS3EndpointProvider().resolveEndpoint(params)
        }
}
```

`endpointUrl` et `endpointProvider`

L'exemple de programme suivant illustre l'interaction entre les `endpointProvider` paramètres `endpointUrl` et. Il s'agit d'un cas d'utilisation avancé.

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.endpoints.DefaultS3EndpointProvider
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointParameters
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointProvider
import aws.smithy.kotlin.runtime.client.endpoints.Endpoint

fun main() = runBlocking {
    S3Client.fromEnvironment {
        endpointUrl = Url.parse("https://example.endpoint")
        endpointProvider = CustomS3EndpointProvider()
    }.use { s3 ->
        // ...
    }
}

class CustomS3EndpointProvider : S3EndpointProvider {
```

```
override suspend fun resolveEndpoint(params: S3EndpointParameters) {
    // The resolved string value of the endpointUrl set in the client above is
    // available here.
    println(params.endpoint)
    // ...
}
}
```

HTTP

Cette section décrit la configuration des paramètres liés au protocole HTTP dans le Kit AWS SDK pour Kotlin

Rubriques

- [Configuration du client HTTP](#)
- [Utiliser un proxy HTTP](#)
- [Intercepteurs HTTP](#)
- [Appliquer une version minimale de TLS](#)

Configuration du client HTTP

Par défaut, Kit AWS SDK pour Kotlin utilise un client HTTP basé sur [OkHttp](#). Vous pouvez remplacer le client HTTP et sa configuration en fournissant un client configuré de manière explicite.

Note

Par défaut, chaque client de service utilise sa propre copie d'un client HTTP. Si vous utilisez plusieurs services dans votre application, vous souhaitez peut-être créer un seul client HTTP et le partager entre tous les clients de service.

Configuration de base

Lorsque vous configurez un client de service, vous pouvez configurer le type de moteur par défaut. Le SDK gère le moteur client HTTP qui en résulte et le ferme automatiquement lorsqu'il n'est plus nécessaire.

L'exemple suivant montre la configuration d'un client HTTP lors de l'initialisation d'un client DynamoDB.

Importations

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import kotlin.time.Duration.Companion.seconds
```

Code

```
DynamoDbClient {
    region = "us-east-2"
    httpClient {
        maxConcurrency = 64u
        connectTimeout = 10.seconds
    }
}.use { ddb ->

    // Perform some actions with Amazon DynamoDB.
}
```

Spécifier un type de moteur HTTP

Pour les cas d'utilisation plus avancés, vous pouvez transmettre un paramètre supplémentaire `httpClient` qui spécifie le type de moteur. Ainsi, vous pouvez définir des paramètres de configuration propres à ce type de moteur.

L'exemple suivant indique [OkHttpEngine](#) ce que vous pouvez utiliser pour configurer la [maxConcurrencyPerHost](#) propriété.

Importations

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.smithy.kotlin.runtime.http.engine.okhttp.OkHttpEngine
```

Code

```
DynamoDbClient {
    region = "us-east-2"
    httpClient(OkHttpEngine) { // The first parameter specifies the HTTP engine type.
```

```
// The following parameter is generic HTTP configuration available in any
engine type.
maxConcurrency = 64u

// The following parameter is OkHttp-specific configuration.
maxConcurrencyPerHost = 32u
}
}.use { ddb ->

// Perform some actions with Amazon DynamoDB.
}
```

Les valeurs possibles pour le type de moteur sont `OkHttpEngine` [OkHttp4Engine](#), et [CrtHttpEngine](#).

Pour utiliser des paramètres de configuration spécifiques à un moteur HTTP, vous devez ajouter le moteur en tant que dépendance au moment de la compilation. Pour le `OkHttpEngine`, vous ajoutez la dépendance suivante à l'aide de Gradle.

(Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.)

```
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
implementation("aws.smithy.kotlin:http-client-engine-okhttp")
```

Pour le `CrtHttpEngine`, ajoutez la dépendance suivante.

```
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
implementation("aws.smithy.kotlin:http-client-engine-crt")
```

Utilisation de la `OkHttp4Engine`

Utilisez le `OkHttp4Engine` si vous ne pouvez pas utiliser la valeur par défaut `OkHttpEngine`. Le [GitHub référentiel smithy-kotlin](#) contient des informations sur la façon dont vous configurez et utilisez le `OkHttp4Engine`.

Utiliser un client HTTP explicite

Lorsque vous utilisez un client HTTP explicite, vous êtes responsable de sa durée de vie, y compris de sa fermeture lorsque vous n'en avez plus besoin. Un client HTTP doit vivre au moins aussi longtemps que n'importe quel client de service qui l'utilise.

L'exemple de code suivant montre le code qui permet au client HTTP de rester actif lorsqu'`DynamoDbClient` est actif. La [use](#) fonction s'assure que le client HTTP se ferme correctement.

Importations

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.smithy.kotlin.runtime.http.engine.okhttp.OkHttpClient
import kotlin.time.Duration.Companion.seconds
```

Code

```
OkHttpClient {
    maxConcurrency = 64u
    connectTimeout = 10.seconds
}.use { okHttpClient ->

    DynamoDbClient {
        region = "us-east-2"
        httpClient = okHttpClient
    }.use { ddb ->
        {
            // Perform some actions with Amazon DynamoDB.
        }
    }
}
```

Utiliser un proxy HTTP

Pour accéder AWS via des serveurs proxy à l'aide du Kit AWS SDK pour Kotlin, vous pouvez configurer les propriétés du système JVM ou des variables d'environnement. Si les deux sont fournis, les propriétés du système JVM sont prioritaires.

Utiliser les propriétés du système JVM

Le SDK recherche les propriétés du système JVM `https.proxyHosthttps.proxyPort`, et `http.nonProxyHosts` Pour plus d'informations sur ces propriétés courantes des systèmes JVM, consultez la section [Mise en réseau et proxies](#) dans la documentation Java.

```
java -Dhttps.proxyHost=10.15.20.25 -Dhttps.proxyPort=1234 -
Dhttp.nonProxyHosts=localhost|api.example.com MyApplication
```


Utiliser des variables d'environnement

Le SDK recherche les variables `https_proxy` et `http_proxy`, et `no_proxy` (ainsi que les versions en majuscules de chacune).

```
export http_proxy=http://10.15.20.25:1234
export https_proxy=http://10.15.20.25:5678
export no_proxy=localhost,api.example.com
```

Utiliser un proxy sur les EC2 instances

Si vous configurez un proxy sur une EC2 instance lancée avec un rôle IAM attaché, veillez à exempter l'adresse utilisée pour accéder aux [métadonnées de l'instance](#). Pour ce faire, définissez la propriété ou la variable `no_proxy` du système `http.nonProxyHosts` JVM sur l'adresse IP du service de métadonnées d'instance, qui est `169.254.169.254`. Cette adresse ne varie pas.

```
export no_proxy=169.254.169.254
```

Intercepteurs HTTP

Vous pouvez utiliser des intercepteurs pour vous connecter à l'exécution des demandes et réponses d'API. Les intercepteurs sont des mécanismes ouverts dans lesquels le SDK appelle le code que vous écrivez pour injecter du comportement dans le cycle de vie de la demande/réponse. De cette façon, vous pouvez modifier une demande en cours de vol, le traitement d'une demande de débogage, consulter les exceptions, etc.

L'exemple suivant montre un intercepteur simple qui ajoute un en-tête supplémentaire à toutes les demandes sortantes avant que la boucle de nouvelle tentative ne soit entrée.

```
class AddHeader(
    private val key: String,
    private val value: String
) : HttpInterceptor {
    override suspend fun modifyBeforeRetryLoop(context:
    ProtocolRequestInterceptorContext<Any, HttpRequest>): HttpRequest {
        val httpReqBuilder = context.protocolRequest.toBuilder()
        httpReqBuilder.headers[key] = value
        return httpReqBuilder.build()
    }
}
```

```
}
```

Pour plus d'informations et pour connaître les hooks d'interception disponibles, consultez l'[interface `Interceptor`](#).

Enregistrement de l'intercepteur

Vous enregistrez des intercepteurs lorsque vous créez un client de service ou lorsque vous remplacez la configuration pour un ensemble spécifique d'opérations.

Intercepteur pour toutes les opérations des clients de service

Le code suivant ajoute une `AddHeader` instance à la propriété `interceptors` du générateur. Cet ajout ajoute l'`x-foo-version` en-tête à toutes les opérations avant l'entrée dans la boucle de nouvelle tentative.

```
val s3 = S3Client.fromEnvironment {
    interceptors += AddHeader("x-foo-version", "1.0")
}

// All service operations invoked using 's3' will have the header appended.
s3.listBuckets { ... }
s3.listObjectsV2 { ... }
```

Intercepteur pour des opérations spécifiques uniquement

En utilisant l'`withConfig` extension, vous pouvez [remplacer la configuration du client de service](#) pour une ou plusieurs opérations pour n'importe quel client de service. Grâce à cette fonctionnalité, vous pouvez enregistrer des intercepteurs supplémentaires pour un sous-ensemble d'opérations.

L'exemple suivant remplace la configuration de l'`s3` instance pour les opérations au sein de l'`use` extension. Les opérations appelées `s3Scoped` contiennent à la fois les en-têtes `x-foo-version` et les `x-bar-version` en-têtes.

```
// 's3' instance created in the previous code snippet.
s3.withConfig {
    interceptors += AddHeader("x-bar-version", "3.7")
}.use { s3Scoped ->
    // All service operations invoked using 's3Scoped' trigger interceptors
    // that were registered when the client was created and any added in the
    // withConfig { ... } extension.
```

```
}
```

Appliquer une version minimale de TLS

Avec le Kit AWS SDK pour Kotlin, vous pouvez configurer la version minimale de TLS lorsque vous vous connectez aux points de terminaison du service. Le SDK propose différentes options de configuration. Par ordre de priorité le plus élevé ou le plus faible, les options sont les suivantes :

- Configuration explicite du moteur HTTP
- Définissez la propriété du système `sdk.minTls JVM`
- Définissez la variable d'`SDK_MIN_TLS` environnement

Configuration du moteur HTTP

Lorsque vous spécifiez un moteur HTTP autre que celui par défaut pour un client de service, vous pouvez définir le `tlsContext.minVersion` champ.

L'exemple suivant configure le moteur HTTP et tout client de service qui l'utilise pour utiliser au minimum le protocole TLS v1.2.

```
DynamoDbClient {
    region = "us-east-2"
    httpClient {
        tlsContext {
            minVersion = TlsVersion.TLS_1_2
        }
    }
}.use { ddb ->

    // Perform some actions with Amazon DynamoDB.
}
```

Définissez la propriété du système `sdk.minTls JVM`

Vous pouvez définir la propriété du système `sdk.minTls JVM`. Lorsque vous lancez une application dont les propriétés système sont définies, tous les moteurs HTTP créés par le Kit AWS SDK pour Kotlin utilisent par défaut la version minimale de TLS spécifiée. Toutefois, vous pouvez le remplacer explicitement dans la configuration du moteur HTTP. Les valeurs autorisées sont les suivantes :

- `TLS_1_0`

- TLS_1_1
- TLS_1_2
- TLS_1_3

Définissez la variable d'**SDK_MIN_TLS**environnement

Vous pouvez définir la variable d'**SDK_MIN_TLS**environnement. Lorsque vous lancez une application avec la variable d'environnement définie, tous les moteurs HTTP créés par le Kit AWS SDK pour Kotlin utilisent la version minimale de TLS spécifiée, sauf si une autre option les remplace.

Les valeurs autorisées sont les suivantes :

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

Nouvelle tentative

Appels renvoyant de Services AWS temps en temps des exceptions inattendues. Certains types d'erreurs, tels que les erreurs de régulation ou les erreurs transitoires, peuvent réussir si l'appel est relancé.

Cette page explique comment configurer les nouvelles tentatives automatiques avec le Kit AWS SDK pour Kotlin.

Configuration de nouvelle tentative par défaut

Par défaut, chaque client de service est automatiquement configuré avec une [stratégie de nouvelle tentative standard](#). La configuration par défaut essaie un appel qui échoue jusqu'à trois fois (la première tentative plus deux tentatives). Le délai entre chaque appel est configuré avec un recul exponentiel et une instabilité aléatoire afin d'éviter les tempêtes de nouvelles tentatives. Cette configuration fonctionne dans la majorité des cas d'utilisation, mais peut s'avérer inappropriée dans certaines circonstances, comme dans le cas des systèmes à haut débit.

Le SDK tente de réessayer uniquement en cas d'erreur réessayable. Parmi les erreurs réessayables, citons les délais d'expiration des sockets, le ralentissement côté service, la simultanéité ou les

échecs de verrouillage optimistes, ainsi que les erreurs de service transitoires. Les paramètres manquants ou non valides, les erreurs d'authentification/de sécurité et les exceptions de mauvaise configuration ne sont pas considérés comme réessayables.

Vous pouvez personnaliser la stratégie de réessai standard en définissant le nombre maximal de tentatives, de délais et d'interruptions, ainsi que la configuration du bucket de jetons.

Nombre maximum de tentatives

Vous pouvez personnaliser le nombre maximal de tentatives par défaut (3) dans le [bloc `retryStrategy DSL`](#) lors de la construction du client.

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        maxAttempts = 5
    }
}
```

Avec le client de service DynamoDB illustré dans l'extrait précédent, le SDK essaie les appels d'API qui échouent jusqu'à cinq fois (la première tentative plus quatre nouvelles tentatives).

Vous pouvez désactiver complètement les tentatives automatiques en fixant le nombre maximum de tentatives à une, comme indiqué dans l'extrait suivant.

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        maxAttempts = 1 // The SDK makes no retries.
    }
}
```

Retards et retards

Si une nouvelle tentative est nécessaire, la stratégie de nouvelle tentative par défaut attend avant d'effectuer la prochaine tentative. Le délai de la première tentative est faible, mais il augmente de façon exponentielle lors des tentatives ultérieures. Le délai maximal est plafonné afin qu'il ne devienne pas trop important.

Enfin, une instabilité aléatoire est appliquée aux délais entre toutes les tentatives. L'instabilité contribue à atténuer les effets des grandes flottes qui peuvent provoquer des tempêtes de nouvelles

tentatives. (Consultez ce billet de [blog sur AWS l'architecture](#) pour une discussion plus approfondie sur le recul et l'instabilité exponentiels.)

Les paramètres de délai sont configurables dans le [bloc `delayProvider DSL`](#).

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        delayProvider {
            initialDelay = 100.milliseconds
            maxBackoff = 5.seconds
        }
    }
}
```

Avec la configuration présentée dans l'extrait précédent, le client retarde la première tentative jusqu'à 100 millisecondes. Le délai maximum entre chaque nouvelle tentative est de 5 secondes.

Les paramètres suivants sont disponibles pour le réglage des délais et du ralentissement.

Paramètre	Valeur par défaut	Description
<code>initialDelay</code>	10 millisecondes	Délai maximal pour la première tentative. Lorsque l'instabilité est appliquée, le délai réel peut être moindre.
<code>jitter</code>	1,0 (gigue complète)	Amplitude maximale permettant de réduire de manière aléatoire le délai calculé. La valeur par défaut de 1,0 signifie que le délai calculé peut être réduit à n'importe quel montant jusqu'à 100 % (par exemple, jusqu'à 0). Une valeur de 0,5 signifie que le délai calculé peut être réduit de moitié au maximum. Ainsi, un délai maximum de 10 ms pourrait être réduit à une

Paramètre	Valeur par défaut	Description
		<p>valeur comprise entre 5 ms et 10 ms. Une valeur de 0,0 signifie qu'aucune instabilité n'est appliquée.</p> <div data-bbox="1068 432 1507 890" style="border: 1px solid #f08080; border-radius: 10px; padding: 10px;"><p>⚠ Important</p><p># La configuration Jitter est une fonctionnalité avancée. La personnalisation de ce comportement n'est généralement pas recommandée.</p></div>
maxBackoff	20 secondes	<p>Le délai maximal à appliquer à toute tentative. La définition de cette valeur limite la croissance exponentielle qui se produit entre les tentatives suivantes et évite que le maximum calculé ne soit trop élevé. Ce paramètre limite le délai calculé avant l'application de la gigue. Si elle est appliquée, la gigue peut encore réduire le délai.</p>

Paramètre	Valeur par défaut	Description
<code>scaleFactor</code>	1.5	<p>Base exponentielle selon laquelle les délais maximaux ultérieurs seront augmentés. Par exemple, étant donné un <code>initialDelay</code> de 10 ms et un <code>scaleFactor</code> de 1,5, les délais maximaux suivants seraient calculés :</p> <ul style="list-style-type: none">• Réessayer 1 : $10 \text{ ms} \times 1,5 = 10 \text{ ms}$• Réessai 2 : $10 \text{ ms} \times 1,5^1 = 15 \text{ ms}$• Réessai 3 : $10 \text{ ms} \times 1,5^2 = 22,5 \text{ ms}$• Réessai 4 : $10 \text{ ms} \times 1,5^3 = 33,75 \text{ ms}$ <p>Lorsque l'instabilité est appliquée, le montant réel de chaque retard peut être moindre.</p>

Réessayez le bucket de jetons

Vous pouvez modifier davantage le comportement de la stratégie de nouvelle tentative standard en ajustant la configuration du bucket de jetons par défaut. Le bucket de jetons de réessai permet de réduire le nombre de tentatives qui ont moins de chances de réussir ou dont la résolution peut prendre plus de temps, telles que les échecs liés au délai imparti ou à la limitation des délais.

⚠ Important

La configuration du bucket à jetons est une fonctionnalité avancée. La personnalisation de ce comportement n'est généralement pas recommandée.

Chaque nouvelle tentative (y compris éventuellement la tentative initiale) réduit une partie de la capacité du bucket de jetons. Le montant décrétement dépend du type de tentative. Par exemple, il peut être peu coûteux de réessayer des erreurs transitoires, mais une nouvelle tentative d'expiration ou de limitation des erreurs peut s'avérer plus coûteuse.

Une tentative réussie rétablit la capacité du compartiment. Le godet ne doit pas être incrémenté au-delà de sa capacité maximale ni décrétement en dessous de zéro.

En fonction de la valeur du `useCircuitBreakerMode` paramètre, les tentatives de réduction de la capacité en dessous de zéro entraînent l'un des résultats suivants :

- Si le paramètre est `VRAI`, une exception est déclenchée. Par exemple, si trop de tentatives ont eu lieu et qu'il est peu probable que d'autres tentatives aboutissent.
- Si le paramètre est `FALSE`, il y a un délai, par exemple, jusqu'à ce que le compartiment ait de nouveau une capacité suffisante.

Les paramètres du bucket de jetons sont configurables dans le [bloc `tokenBucket` DSL](#) :

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        tokenBucket {
            maxCapacity = 100
            refillUnitsPerSecond = 2
        }
    }
}
```

Les paramètres suivants sont disponibles pour régler le bucket de jetons `Retry` :

Paramètre	Valeur par défaut	Description
<code>initialTryCost</code>	0	Le montant à déduire du bucket lors des premières

Paramètre	Valeur par défaut	Description
		tentatives. La valeur par défaut de 0 signifie qu'aucune capacité ne sera décrétementée et que les tentatives initiales ne seront donc ni interrompues ni retardées.
<code>initialTrySuccessIncrement</code>	1	Le montant à augmenter lorsque la première tentative a été couronnée de succès.
<code>maxCapacity</code>	500	Capacité maximale du bucket de jetons. Le nombre de jetons disponibles ne peut pas dépasser ce nombre.
<code>refillUnitsPerSecond</code>	0	La quantité de capacité ajoutée au godet chaque seconde. Une valeur de 0 signifie qu'aucune capacité n'est automatiquement ajoutée à nouveau. (Par exemple, seules les tentatives réussies permettent d'augmenter la capacité). Une valeur de 0 <code>useCircuitBreakerMode</code> doit être vraie.
<code>retryCost</code>	5	Le montant à déduire du compartiment en cas de tentative consécutive à un échec temporaire. Le même montant est réincrémenté dans le compartiment si la tentative est réussie.

Paramètre	Valeur par défaut	Description
<code>timeoutRetryCost</code>	10	Le montant à déduire du compartiment en cas de tentative consécutive à un délai d'expiration ou à une défaillance de la régulation. Le même montant est réincrémenté dans le compartiment si la tentative est réussie.
<code>useCircuitBreakerMode</code>	TRUE	Détermine le comportement lorsqu'une tentative de réduction de la capacité entraîne une chute de la capacité du compartiment en dessous de zéro. Lorsque la valeur est TRUE, le bucket de jetons génère une exception indiquant qu'il n'existe plus de capacité de nouvelle tentative. Lorsque la valeur est définie sur FALSE, le bucket de jetons retarde la tentative jusqu'à ce que la capacité soit remplie en quantité suffisante.

Tentatives adaptatives

Comme alternative à la stratégie de nouvelle tentative standard, la stratégie de nouvelle tentative adaptative est une approche avancée qui recherche le taux de demandes idéal afin de minimiser les erreurs de limitation.

⚠ Important

Les tentatives adaptatives sont un mode de nouvelle tentative avancé. L'utilisation de cette stratégie de nouvelle tentative n'est généralement pas recommandée.

Les tentatives adaptatives incluent toutes les fonctionnalités des tentatives standard. Il ajoute un limiteur de débit côté client qui mesure le taux de demandes limitées par rapport aux demandes non limitées. Cela limite également le trafic pour tenter de rester dans une bande passante sûre, ce qui ne provoque idéalement aucune erreur de régulation.

Le tarif s'adapte en temps réel à l'évolution des conditions de service et des modèles de trafic et peut augmenter ou diminuer le taux de trafic en conséquence. Surtout, le limiteur de débit peut retarder les premières tentatives dans les scénarios à fort trafic.

Vous sélectionnez la stratégie de nouvelle tentative adaptative en fournissant un paramètre supplémentaire à la `retryStrategy` méthode. Les paramètres du limiteur de débit sont configurables dans le [bloc `rateLimiter` DSL](#).

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy(AdaptiveRetryStrategy) {
        maxAttempts = 10
        rateLimiter {
            minFillRate = 1.0
            smoothing = 0.75
        }
    }
}
```

📌 Note

La stratégie de nouvelle tentative adaptative suppose que le client travaille sur une seule ressource (par exemple, une table DynamoDB ou un compartiment Amazon S3).

Si vous utilisez un seul client pour plusieurs ressources, les ralentissements ou les pannes associés à une ressource entraînent une latence accrue et des défaillances lorsque le client accède à toutes les autres ressources. Lorsque vous utilisez la stratégie de nouvelle tentative adaptative, nous vous recommandons d'utiliser un seul client pour chaque ressource.

Observabilité

L'observabilité est la mesure dans laquelle l'état actuel d'un système peut être déduit des données qu'il émet. Les données émises sont communément appelées télémétrie.

Ils Kit AWS SDK pour Kotlin peuvent fournir les trois signaux de télémétrie courants : métriques, traces et logs. Vous pouvez connecter un [TelemetryProvider](#) pour envoyer des données de télémétrie à un backend d'observabilité (tel que [AWS X-Ray Amazon CloudWatch](#)), puis agir en conséquence.

Par défaut, seule la journalisation est activée et les autres signaux de télémétrie sont désactivés dans le SDK. Cette rubrique explique comment activer et configurer la sortie de télémétrie.

Important

`TelemetryProvider` est actuellement une API expérimentale qui doit être activée pour être utilisée.

Configurez un `TelemetryProvider`

Vous pouvez configurer un `TelemetryProvider` dans votre application globalement pour tous les clients de service ou pour des clients individuels. Les exemples suivants utilisent une `getConfiguredProvider()` fonction hypothétique pour illustrer les opérations de l'`TelemetryProviderAPI`. La [the section called “Fournisseurs de services de télémétrie”](#) section décrit les informations relatives aux implémentations fournies par le SDK. Si un fournisseur n'est pas pris en charge, vous pouvez implémenter votre propre support ou [ouvrir une demande de fonctionnalité sur GitHub](#).

Configuration du fournisseur de télémétrie global par défaut

Par défaut, chaque client de service tente d'utiliser le fournisseur de télémétrie disponible dans le monde entier. De cette façon, vous pouvez définir le fournisseur une seule fois, et tous les clients l'utiliseront. Cela ne doit être fait qu'une seule fois, avant d'instancier un client de service.

Pour utiliser le fournisseur de télémétrie global, mettez d'abord à jour les dépendances de votre projet pour ajouter le module de télémétrie par défaut, comme indiqué dans l'extrait de code Gradle suivant.

(Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.)

```
dependencies {
    implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
    implementation("aws.smithy.kotlin:telemetry-defaults")
    ...
}
```

Définissez ensuite le fournisseur de télémétrie global avant de créer un client de service, comme indiqué dans le code suivant.

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.smithy.kotlin.runtime.telemetry.GlobalTelemetryProvider
import kotlinx.coroutines.runBlocking

fun main() = runBlocking {
    val myTelemetryProvider = getConfiguredProvider()
    GlobalTelemetryProvider.set(myTelemetryProvider)

    S3Client.fromEnvironment().use { s3 ->
        ...
    }
}

fun getConfiguredProvider(): TelemetryProvider {
    TODO("TODO - configure a provider")
}
```

Configuration d'un fournisseur de télémétrie pour un client de service spécifique

Vous pouvez configurer un client de service individuel auprès d'un fournisseur de télémétrie spécifique (autre que le fournisseur global). Voici un exemple :

```
import aws.sdk.kotlin.services.s3.S3Client
import kotlinx.coroutines.runBlocking

fun main() = runBlocking {
    S3Client.fromEnvironment{
        telemetryProvider = getConfiguredProvider()
    }.use { s3 ->
        ...
    }
}
```

```
fun getConfiguredProvider(): TelemetryProvider {
    TODO("TODO - configure a provider")
}
```

Métriques

Le tableau suivant répertorie les mesures de télémétrie émises par le SDK. [Configurez un fournisseur de télémétrie](#) pour rendre les métriques observables.

Quels indicateurs sont émis ?

Nom de la métrique	Unités	Type	Attributs	Description
smithy.client.call.duration	s	Histogramme	rpc.service, rpc.method	Durée totale de l'appel (y compris les nouvelles tentatives)
smithy.client.call.attempts	{tentative}	Monoton Counter	rpc.service, rpc.method	Le nombre de tentatives pour une opération individuelle
smithy.client.call.errors	{erreur}	Monoton Counter	rpc.service, rpc.method, exception.type	Le nombre d'erreurs associées à une opération
smithy.client.call.attempt_duration	s	Histogramme	rpc.service, rpc.method	Le temps nécessaire pour se connecter au service, envoyer la demande et récupérer le code d'état HTTP et les en-têtes (y compris le temps passé dans la file d'attente avant d'être envoyé)
smithy.client.call.resolve_endpoint_duration	s	Histogramme	rpc.service, rpc.method	Le temps nécessaire pour résoudre un point de terminaison (résolveur de point de terminaison, pas DNS) pour la demande
smithy.client.call.serialization_duration	s	Histogramme	rpc.service, rpc.method	Le temps nécessaire pour sérialiser le corps d'un message

Nom de la métrique	Unités	Type	Attributs	Description
smithy.client.serialization_duration	s	Histogramme	rpc.service, rpc.method	Le temps nécessaire pour désérialiser le corps d'un message
smithy.client.auth.signing_duration	s	Histogramme	rpc.service, rpc.method, auth.scheme_id	Le temps nécessaire pour signer une demande
smithy.client.call.resolve_identity_duration	s	Histogramme	rpc.service, rpc.method, auth.scheme_id	Le temps nécessaire pour acquérir une identité (telle qu'un AWS identifiant ou un jeton porteur) auprès d'un fournisseur d'identité
smithy.client.http.connections.acquire_duration	s	Histogramme		Le temps nécessaire à une demande pour établir une connexion
smithy.client.http.connections.limit	{connexion}	[Asynchrone] UpDownCounter		Nombre maximal de connexions ouvertes autorisées/configurées pour le client HTTP
smithy.client.http.connections.usage	{connexion}	[Asynchrone] UpDownCounter	état : inactif acquis	État actuel du pool de connexions
smithy.client.http.connections.uptime	s	Histogramme		Durée pendant laquelle une connexion a été ouverte

Nom de la métrique	Unités	Type	Attributs	Description
smithy.client.http.requests.usage	{demande}	[Asynchrone] UpDownCounter	état : en file d'attente en vol	État actuel de la simultanéité des demandes du client HTTP
smithy.client.http.requests.queued_duration	s	Histogramme		Le temps passé par une requête en file d'attente et en attente d'être exécutée par le client HTTP
smithy.client.http.bytes_sent	Par	Monoton Counter	adresse du serveur	Le nombre total d'octets envoyés par le client HTTP
smithy.client.http.bytes_received	Par	Monoton Counter	adresse du serveur	Le nombre total d'octets reçus par le client HTTP

Les descriptions des colonnes sont les suivantes :

- Nom de la métrique : nom de la métrique émise.
- Unités : unité de mesure de la métrique. Les unités sont données dans la notation [UCUM](#) sensible aux majuscules et minuscules (« c/s »).
- Type : type d'instrument utilisé pour capturer la métrique.
- Description —Description de ce que mesure la métrique.
- Attributs : ensemble d'attributs (dimensions) émis avec la métrique.

Journalisation

Kit AWS SDK pour Kotlin Configure un enregistreur compatible [SLF4J](#) comme valeur par défaut du fournisseur `LoggerProvider` de télémétrie. Avec SLF4J, qui est une couche d'abstraction, vous pouvez utiliser n'importe lequel des nombreux systèmes de journalisation au moment de l'exécution. [Les systèmes de journalisation pris en charge incluent Java Logging APIs, Log4j 2 et Logback.](#)

⚠ Warning

Nous vous recommandons de n'utiliser l'enregistrement des câbles qu'à des fins de débogage. (L'enregistrement des câbles est discuté ci-dessous.) Désactivez-le dans vos environnements de production car il peut enregistrer des données sensibles telles que les adresses e-mail, les jetons de sécurité, les clés d'API, les mots de passe et AWS Secrets Manager les secrets. L'enregistrement des connexions enregistre l'intégralité de la demande ou de la réponse sans chiffrement, même pour un appel HTTPS.

Pour les demandes volumineuses (telles que le téléchargement d'un fichier sur Amazon S3) ou les réponses, l'enregistrement détaillé des connexions peut également avoir un impact significatif sur les performances de votre application.

Exemple de configuration de journalisation Log4j 2

Bien que n'importe quelle bibliothèque de journaux SLF4J compatible puisse être utilisée, cet exemple active la sortie des journaux à partir du SDK dans les programmes JVM utilisant Log4j 2 :

Dépendances de Gradle

(Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.)

```
implementation("org.apache.logging.log4j:log4j-slf4j2-impl:X.Y.Z")
```

Fichier de configuration Log4j 2

Créez un fichier nommé `log4j2.xml` dans votre ressources répertoire (par exemple, `<project-dir>/src/main/resources`). Ajoutez la configuration XML suivante au fichier :

```
<Configuration status="ERROR">
  <Appenders>
    <Console name="Out">
      <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} %-5p %c:%L %X - %encode{%m}
{CRLF}%n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Root level="info">
      <AppenderRef ref="Out"/>
    </Root>
  </Loggers>
</Configuration>
```

```
</Loggers>  
</Configuration>
```

Cette configuration inclut le %X spécificateur dans l'attribut `pattern` qui active la journalisation MDC (contexte de diagnostic mappé).

Le SDK ajoute les éléments MDC suivants pour chaque opération.

`rpc`

Le nom du RPC invoqué, par exemple `S3.GetObject`.

`sdkInvocationId`

Un identifiant unique attribué par le client du service pour l'opération. L'ID met en corrélation tous les événements de journalisation liés à l'invocation d'une seule opération.

Spécifier le mode journal pour les messages au niveau du fil

Par défaut, les messages Kit AWS SDK pour Kotlin filaires ne sont pas enregistrés car ils peuvent contenir des données sensibles provenant de demandes et de réponses d'API. Cependant, vous avez parfois besoin de ce niveau de détail à des fins de débogage.

Avec le SDK Kotlin, vous pouvez définir un mode journal dans le code ou utiliser les paramètres d'environnement pour activer les messages de débogage pour les éléments suivants :

- Requêtes HTTP
- Réponses HTTP

Le mode journal est soutenu par un champ de bits où chaque bit est un indicateur (mode) et les valeurs sont additives. Vous pouvez combiner un mode de demande et un mode de réponse.

Définir le mode journal dans le code

Pour activer la journalisation supplémentaire, définissez la `LogMode` propriété lorsque vous créez un client de service.

L'exemple suivant montre comment activer l'enregistrement des demandes (avec le corps) et de la réponse (sans le corps).

```
import aws.smithy.kotlin.runtime.client.LogMode
```

```
// ...  
  
val client = DynamoDbClient {  
    // ...  
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse  
}
```

Une valeur de mode journal définie lors de la construction du client de service remplace toute valeur de mode journal définie dans l'environnement.

Définir le mode journal depuis l'environnement

Pour définir un mode journal global pour tous les clients de service qui ne sont pas explicitement configurés dans le code, utilisez l'une des méthodes suivantes :

- Propriété du système JVM : `sdk.logMode`
- Variable d'environnement : `SDK_LOG_MODE`

Les valeurs suivantes, qui ne distinguent pas les majuscules des minuscules, sont disponibles :

- `LogRequest`
- `LogRequestWithBody`
- `LogResponse`
- `LogResponseWithBody`

Pour créer un mode journal combiné à l'aide des paramètres de l'environnement, vous devez séparer les valeurs par un symbole en forme de tube (|).

Par exemple, les exemples suivants définissent le même mode de journalisation que dans l'exemple précédent.

```
# Environment variable.  
export SDK_LOG_MODE=LogRequestWithBody|LogResponse
```

```
# JVM system property.  
java -Dsdk.logMode=LogRequestWithBody|LogResponse ...
```

Note

Vous devez également configurer un enregistreur SLF4 J compatible et définir le niveau de journalisation sur DEBUG pour activer la journalisation au niveau du fil.

Fournisseurs de services de télémétrie

Le SDK prend actuellement en charge [OpenTelemetry](#)(OTel) en tant que fournisseur. Le SDK pourrait proposer des fournisseurs de télémétrie supplémentaires à l'avenir.

Rubriques

- [Configuration du fournisseur OpenTelemetry de télémétrie basé](#)

Configuration du fournisseur OpenTelemetry de télémétrie basé

Le SDK pour Kotlin fournit une implémentation de l'`TelemetryProvider` interface soutenue par OpenTelemetry

Prérequis

Mettez à jour les dépendances de votre projet pour ajouter le OpenTelemetry fournisseur, comme indiqué dans l'extrait de code Gradle suivant. Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.

```
dependencies {
    implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
    implementation(platform("io.opentelemetry.instrumentation:opentelemetry-
instrumentation-bom:X.Y.Z"))
    implementation("aws.smithy.kotlin:telemetry-provider-otel")

    // OPTIONAL: If you use log4j, the following entry enables the ability to export
    logs through OTel.
    runtimeOnly("io.opentelemetry.instrumentation:opentelemetry-log4j-appender-2.17")
}
```

Configuration du kit SDK

Le code suivant configure un client de service à l'aide du fournisseur de OpenTelemetry télémétrie.

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.smithy.kotlin.runtime.telemetry.otel.OpenTelemetryProvider
import io.opentelemetry.api.GlobalOpenTelemetry
import kotlinx.coroutines.runBlocking

fun main() = runBlocking {
    val otelProvider = OpenTelemetryProvider(GlobalOpenTelemetry.get())

    S3Client.fromEnvironment().use { s3 ->
        telemetryProvider = otelProvider
        ...
    }
}
```

Note

Une discussion sur la configuration du OpenTelemetry SDK n'entre pas dans le cadre de ce guide. La [documentation OpenTelemetry Java](#) contient des informations de configuration sur les différentes approches : [manuellement](#), automatiquement via l'[agent Java](#) ou le [collecteur](#) (facultatif).

Ressources

Les ressources suivantes sont disponibles pour vous aider à démarrer OpenTelemetry.

- [AWS Page d'accueil de Distro for OpenTelemetry](#) - AWS OTe L Distro
- [aws-otel-java-instrumentation](#)- AWS Bibliothèque d'instrumentation Distro pour OpenTelemetry Java
- [aws-otel-lambda](#)- couches OpenTelemetry Lambda AWS gérées
- [aws-otel-collector](#)- AWS Distro pour collectionneur OpenTelemetry
- [AWS Meilleures pratiques d'observabilité - Bonnes pratiques](#) générales en matière d'observabilité spécifiques à AWS

Remplacer la configuration du client de service

Une fois qu'un [client de service est créé](#), celui-ci utilise une configuration fixe pour toutes les opérations. Cependant, il se peut que vous deviez parfois remplacer la configuration pour une ou plusieurs opérations spécifiques.

Chaque client de service possède une `withConfig` extension qui vous permet de modifier une copie de la configuration existante. L'`withConfig` extension renvoie un nouveau client de service avec une configuration modifiée. Le client d'origine existe indépendamment et utilise sa configuration d'origine.

L'exemple suivant montre la création d'une `S3Client` instance qui appelle deux opérations.

```
val s3 = S3Client.fromEnvironment {
    logMode = LogMode.LogRequest
    region = "us-west-2"
    // ...other configuration settings...
}

s3.listBuckets { ... }
s3.listObjectsV2 { ... }
```

L'extrait suivant montre comment remplacer la configuration pour une seule opération.

`listObjectsV2`

```
s3.withConfig {
    region = "eu-central-1"
}.use { overriddenS3 ->
    overriddenS3.listObjectsV2 { ... }
}
```

L'opération appelle le `s3` client à utiliser la configuration d'origine spécifiée lors de la création du client. Sa configuration inclut la [journalisation des demandes](#) et `us-west-2` region pour la région.

L'`listObjectsV2` invocation sur le `overriddenS3` client utilise les mêmes paramètres que le `s3` client d'origine, à l'exception de la région, qui est désormais `eu-central-1` utilisée.

Cycle de vie d'un client remplacé

Dans l'exemple précédent, le `s3` client et le `overriddenS3` client sont indépendants l'un de l'autre. Les opérations peuvent être invoquées sur l'un ou l'autre client tant qu'elles restent ouvertes. Chacun

utilise une configuration distincte, mais ils peuvent partager des ressources sous-jacentes (comme un moteur HTTP) à moins que celles-ci ne soient également remplacées.

Vous fermez un client dont la configuration a été remplacée et le client d'origine séparément. Vous pouvez fermer un client dont la configuration a été modifiée avant ou après la fermeture de son client d'origine. À moins que vous n'ayez besoin d'utiliser un client dont la configuration a été remplacée pendant une longue période, nous vous recommandons de terminer son cycle de vie avec cette méthode. La `use` méthode garantit que le client est fermé en cas d'exceptions.

Ressources partagées entre les clients

Lorsque vous créez un client de service à l'aide de `withConfig`, il peut partager des ressources avec le client d'origine. En revanche, lorsque vous créez un client à l'aide de [FromEnvironment](#) ou que vous le [configurez explicitement](#), le client utilise des ressources indépendantes. Les ressources telles que les moteurs HTTP et les fournisseurs d'informations d'identification sont partagées à moins qu'elles ne soient remplacées dans le `withConfig` bloc.

Le cycle de vie de chaque client étant indépendant, les ressources partagées restent ouvertes et utilisables jusqu'à la fermeture du dernier client. Par conséquent, il est important que vous fermiez les clients dont les services ont été annulés lorsque vous n'en avez plus besoin. Cela empêche les ressources partagées de rester ouvertes et de consommer des ressources système telles que la mémoire, la connexion et les cycles du processeur.

L'exemple suivant montre à la fois des ressources partagées et indépendantes.

Les `overriddenS3` clients `s3` et partagent la même instance de fournisseur d'informations d'identification, y compris sa configuration de mise en cache. Les appels effectués en `overriddenS3` réutilisent les informations d'identification si la valeur mise en cache est toujours à jour depuis les appels effectués par le `s3` client.

Le moteur HTTP n'est pas partagé entre les deux clients. Chaque client dispose d'un moteur HTTP indépendant car celui-ci a été remplacé lors de l'appel `withConfig`.

```
val s3 = S3Client.fromEnvironment {
    region = "us-west-2"
    credentialsProvider = CachedCredentialsProvider(CredentialsProviderChain(...))
    httpClientEngine = OkHttpEngine { ... }
}

s3.listBuckets { ... }
```



```
s3.withConfig {  
    httpClientEngine = CrtHttpEngine { ... }  
}.use { overriddenS3 ->  
    overriddenS3.listObjectsV2 { ... }  
}
```

Utilisation de la SDK

Cette section fournit les informations de base requises pour utiliser le Kit AWS SDK pour Kotlin.

Rubriques

- [Faites des demandes](#)
- [Coroutines](#)
- [Opérations de streaming](#)
- [Pagination](#)
- [Programmes d'attente](#)
- [Gestion des erreurs](#)
- [Demandes de pré-signature](#)
- [Résolution des problèmes de FAQs](#)

Faites des demandes

Utilisez un client de service pour envoyer des demandes à un Service AWS. Kit AWS SDK pour Kotlin fournit des langages spécifiques au domaine (DSLs) suivant un [modèle de générateur sécurisé](#) pour créer des demandes. Les structures imbriquées des demandes sont également accessibles via leur DSLs

L'exemple suivant montre comment créer une entrée d'opération Amazon [createTable](#)DynamoDB :

```
val ddb = DynamoDbClient.fromEnvironment()

val req = CreateTableRequest {
    tableName = name
    keySchema = listOf(
        KeySchemaElement {
            attributeName = "year"
            keyType = KeyType.Hash
        },
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }
    )
}
```

```
attributeDefinitions = listOf(
    AttributeDefinition {
        attributeName = "year"
        attributeType = ScalarAttributeType.N
    },
    AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }
)

// You can configure the `provisionedThroughput` member
// by using the `ProvisionedThroughput.Builder` directly:
provisionedThroughput {
    readCapacityUnits = 10
    writeCapacityUnits = 10
}
}

val resp = ddb.createTable(req)
```

DSL Surcharges de l'interface de service

Chaque opération non diffusée sur l'interface du client de service est DSL surchargée, de sorte que vous n'avez pas à créer de demande distincte.

Exemple de création d'un bucket Amazon Simple Storage Service (Amazon S3) avec la fonction surchargée :

```
s3Client.createBucket { // this: CreateBucketRequest.Builder
    bucket = newBucketName
}
```

Cette expression est similaire à :

```
val request = CreateBucketRequest { // this: CreateBucketRequest.Builder
    bucket = newBucketName
}

s3client.createBucket(request)
```

Demandes sans saisie requise

Les opérations qui n'ont pas d'entrées requises peuvent être appelées sans avoir à transmettre un objet de requête. Cela est souvent possible avec des opérations de type liste, telles que l'opération Amazon S3 `listBucketsAPI`.

Par exemple, les trois instructions suivantes sont équivalentes :

```
s3Client.listBuckets(ListBucketsRequest {
    // Construct the request object directly.
})
s3Client.listBuckets {
    // DSL builder without explicitly setting any arguments.
}
s3Client.listBuckets()
```

Coroutines

Kit AWS SDK pour Kotlin II est asynchrone par défaut. Le SDK for Kotlin utilise des `suspend` fonctions pour toutes les opérations, qui sont destinées à être appelées à partir d'une coroutine.

Pour un guide plus détaillé sur les coroutines, consultez la documentation [officielle de Kotlin](#).

Faire des demandes simultanées

Le générateur de coroutine [asynchrone](#) peut être utilisé pour lancer des requêtes simultanées lorsque vous vous souciez des résultats. `asyncrenvoie` un [Deferred](#), qui représente un futur léger et non bloquant qui représente la promesse de fournir un résultat ultérieurement.

Si vous ne vous souciez pas des résultats (uniquement du fait qu'une opération soit terminée), vous pouvez utiliser le générateur de coroutine de [lancement](#). `launch` est conceptuellement similaire à `async`. La différence est que `launch` renvoie un [Job](#) et ne contient aucune valeur résultante, tandis que `async` renvoie un `Deferred`.

Voici un exemple de demande simultanée adressée à Amazon S3 à l'aide de l'[headObject](#) opération visant à obtenir la taille du contenu de deux clés :

```
import kotlinx.coroutines.async
```

```
import kotlinx.coroutines.runBlocking
import kotlin.system.measureTimeMillis
import aws.sdk.kotlin.services.s3.S3Client

fun main(): Unit = runBlocking {

    val s3 = S3Client { region = "us-east-2" }

    val myBucket = "<your-bucket-name-here>"
    val key1 = "<your-object-key-here>"
    val key2 = "<your-second-object-key-here>"

    val resp1 = async {
        s3.headObject{
            bucket = myBucket
            key = key1
        }
    }

    val resp2 = async {
        s3.headObject{
            bucket = myBucket
            key = key2
        }
    }

    val elapsed = measureTimeMillis {
        val totalContentSize = resp1.await().contentLength +
resp2.await().contentLength
        println("content length of $key1 + $key2 = $totalContentSize")
    }

    println("requests completed in $elapsed ms")
}
```

Faire des demandes de blocage

Pour effectuer des appels de service à partir d'un code existant qui n'utilise pas de coroutines et implémente un modèle de thread différent, vous pouvez utiliser le [runBlocking](#) générateur de coroutine. Un exemple de modèle de threading différent est l'utilisation de l'approche traditionnelle

exécuteurs/contrats à terme de Java. Vous devrez peut-être utiliser cette approche si vous mélangez du code ou des bibliothèques Java et Kotlin.

Comme son nom l'indique, ce `runBlocking` générateur lance une nouvelle coroutine et bloque le thread en cours jusqu'à ce qu'il soit terminé.

Warning

`runBlocking` ne doit généralement pas être utilisé à partir d'une coroutine. Il est conçu pour relier le code de blocage normal aux bibliothèques écrites dans un style suspendu (comme dans les fonctions principales et les tests).

Opérations de streaming

Dans le Kit AWS SDK pour Kotlin, les données binaires (flux) sont représentées sous la forme d'un [ByteStream](#)type, qui est un flux d'octets abstrait en lecture seule.

Réponses en streaming

Les réponses contenant un flux binaire (comme l'opération Amazon Simple Storage Service (Amazon [GetObject](#)API3)) sont traitées différemment des autres méthodes. Ces méthodes utilisent une fonction lambda qui gère la réponse plutôt que de la renvoyer directement. Cela limite l'étendue de la réponse à la fonction et simplifie la gestion de la durée de vie à la fois pour l'appelant et pour le temps SDK d'exécution.

Après le retour de la fonction lambda, toutes les ressources, telles que la HTTP connexion sous-jacente, sont libérées. (Le `ByteStream` doit pas être accessible après le retour du lambda et ne doit pas être transmis hors de la fermeture.) Le résultat de l'appel est ce que renvoie le lambda.

L'exemple de code suivant montre la [getObject](#) fonction recevant un paramètre lambda, qui gère la réponse.

```
val s3Client = S3Client.fromEnvironment()
val req = GetObjectRequest { ... }

val path = Paths.get("/tmp/download.txt")

// S3Client.getObject has the following signature:
```

```
// suspend fun <T> getObject(input: GetObjectRequest, block: suspend
  (GetObjectResponse) -> T): T

val contentSize = s3Client.getObject(req) { resp ->
  // resp is valid until the end of the block.
  // Do not attempt to store or process the stream after the block returns.

  // resp.body is of type ByteStream.
  val rc = resp.body?.writeToFile(path)
  rc
}
println("wrote $contentSize bytes to $path")
```

Le `ByteStream` type possède les extensions suivantes pour les méthodes courantes de consommation :

- `ByteStream.writeToFile(file: File): Long`
- `ByteStream.writeToFile(path: Path): Long`
- `ByteStream.toByteArray(): ByteArray`
- `ByteStream.decodeToString(): String`

Tous ces éléments sont définis dans le `aws.smithy.kotlin.runtime.content` package.

Demandes de streaming

Pour fournir un `ByteStream`, il existe également plusieurs méthodes pratiques, notamment les suivantes :

- `ByteStream.fromFile(file: File)`
- `File.asByteStream(): ByteStream`
- `Path.asByteStream(): ByteStream`
- `ByteStream.fromBytes(bytes: ByteArray)`
- `ByteStream.fromString(str: String)`

Tous ces éléments sont définis dans le `aws.smithy.kotlin.runtime.content` package.

L'exemple de code suivant montre l'utilisation de méthodes `ByteStream` pratiques qui fournissent la propriété `body` lors de la création d'un [PutObjectRequest](#):

```
val req = PutObjectRequest {
    ...
    body = ByteStream.fromFile(file)
    // body = ByteStream.fromBytes(byteArray)
    // body = ByteStream.fromString("string")
    // etc
}
```

Pagination

De nombreuses AWS opérations renvoient des résultats paginés lorsque la charge utile est trop importante pour être renvoyée en une seule réponse. Kit AWS SDK pour Kotlin II inclut des [extensions](#) de l'interface client du service qui paginent automatiquement les résultats pour vous. Il suffit d'écrire le code qui traite les résultats.

La pagination est exposée sous forme de [flux <T>](#) afin que vous puissiez tirer parti des transformations idiomatiques de Kotlin pour les collections asynchrones (telles que, et). `map` `filter` `take` Les exceptions sont transparentes, ce qui donne l'impression que la gestion des erreurs s'apparente à un API appel normal, et l'annulation s'inscrit dans le cadre de l'annulation coopérative générale des coroutines. Pour plus d'informations, consultez [les flux et les exceptions](#) de flux dans le guide officiel.

Note

Les exemples suivants utilisent Amazon S3. Cependant, les concepts sont les mêmes pour tout service comportant une ou plusieurs paginées APIs. Toutes les extensions de pagination sont définies dans le `aws.sdk.kotlin.<service>.paginators` package (par exemple `aws.sdk.kotlin.dynamodb.paginators`).

L'exemple de code suivant montre comment traiter la réponse paginée à partir de l'appel de fonction [listObjectsV2Paginated](#).

Importations

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.paginators.listObjectsV2Paginated
import kotlinx.coroutines.flow.*
```


Code

```
val s3 = S3Client.fromEnvironment()
val req = ListObjectsV2Request {
    bucket = "<my-bucket>"
    maxKeys = 1
}

s3.listObjectsV2Paginated(req) // Flow<ListObjectsV2Response>
    .transform { it.contents?.forEach { obj -> emit(obj) } }
    .collect { obj ->
        println("key: ${obj.key}; size: ${obj.size}")
    }
}
```

Programmes d'attente

Les serveurs sont une abstraction côté client utilisée pour interroger une ressource jusqu'à ce qu'un état souhaité soit atteint, ou jusqu'à ce qu'il soit déterminé que la ressource n'entrera pas dans l'état souhaité. Il s'agit d'une tâche courante lorsque vous travaillez avec des services qui sont finalement cohérents, comme Amazon Simple Storage Service (Amazon S3), ou des services qui créent des ressources de manière asynchrone, comme Amazon. EC2

L'écriture d'une logique permettant de vérifier en permanence l'état d'une ressource peut s'avérer fastidieuse et source d'erreurs. L'objectif des serveurs est de faire passer cette responsabilité du code client à celui qui possède une connaissance approfondie des aspects temporels de l' AWS opération. Kit AWS SDK pour Kotlin

Note

Les exemples suivants utilisent Amazon S3. Cependant, les concepts sont les mêmes pour tous ceux Service AWS qui ont un ou plusieurs serveurs définis. Toutes les extensions sont définies dans le `aws.sdk.kotlin.<service>.waiters` package (par exemple `aws.sdk.kotlin.dynamodb.waiters`). Ils suivent également une convention de dénomination standard (`waitUntil<Condition>`).

L'exemple de code suivant montre l'utilisation d'une fonction de serveur qui vous permet d'éviter d'écrire une logique d'interrogation.

Importations

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.waiters.waitUntilBucketExists
```

Code

```
val s3 = S3Client.fromEnvironment()

// This initiates creating an S3 bucket and potentially returns before the bucket
// exists.
s3.createBucket { bucket = "my-bucket" }

// When this function returns, the bucket either exists or an exception
// is thrown.
s3.waitUntilBucketExists { bucket = "my-bucket" }

// The bucket now exists.
```

Note

Chaque méthode d'attente renvoie une Outcome instance qui peut être utilisée pour obtenir la réponse finale correspondant à l'atteinte de la condition souhaitée. Un résultat contient également des détails supplémentaires tels que le nombre de tentatives effectuées pour atteindre l'état souhaité.

Gestion des erreurs

Il est important de comprendre comment et quand les Kit AWS SDK pour Kotlin exceptions déclenchent des exceptions pour créer des applications de haute qualité à l'aide du SDK. Les sections suivantes décrivent les différents cas d'exceptions déclenchées par le SDK et expliquent comment les gérer de manière appropriée.

Exceptions de service

L'exception la plus courante est `AwsServiceException` celle dont héritent toutes les exceptions spécifiques au service (telles que `S3Exception`). Cette exception représente une réponse d'erreur provenant d'un Service AWS. Par exemple, si vous essayez de mettre fin à une EC2 instance Amazon qui n'existe pas, Amazon EC2 renvoie une réponse d'erreur. Les détails de la réponse d'erreur sont inclus dans `AwsServiceException` le message envoyé.

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

Exceptions relatives aux

`ClientException` indique qu'un problème s'est produit dans le code Kit AWS SDK pour Kotlin client, soit lors de la tentative d'envoi d'une demande, AWS soit lors de la tentative d'analyse d'une réponse de AWS. A `ClientException` est généralement plus sévère que A `AwsServiceException` et indique qu'un problème majeur empêche le client de traiter les appels de service à Services AWS. Par exemple, il Kit AWS SDK pour Kotlin renvoie un `ClientException` s'il ne parvient pas à analyser la réponse d'un service.

Métadonnées d'erreur

Chaque exception de service et exception de client possède une `sdkErrorMetadata` propriété. Il s'agit d'un sac de propriétés dactylographié qui peut être utilisé pour récupérer des informations supplémentaires sur l'erreur.

Plusieurs extensions prédéfinies existent directement pour `AwsErrorMetadata` ce type, notamment, mais sans s'y limiter, les suivantes :

- `sdkErrorMetadata.requestId`— l'identifiant unique de la demande
- `sdkErrorMetadata.errorMessage`— le message lisible par l'homme (correspond généralement au `Exception.message`, mais peut contenir plus d'informations si l'exception était inconnue du service)
- `sdkErrorMetadata.protocolResponse`— La réponse brute du protocole

L'exemple suivant montre comment accéder aux métadonnées d'erreur.

```
try {
    s3Client.listBuckets { ... }
} catch (ex: S3Exception) {
    val awsRequestId = ex.sdkErrorMetadata.requestId
    val httpResp = ex.sdkErrorMetadata.protocolResponse as? HttpResponse

    println("requestId was: $awsRequestId")
    println("http status code was: ${httpResp?.status}")
}
```

```
}
```

Demandes de pré-signature

Vous pouvez présigner les demandes pour certaines AWS API opérations afin qu'un autre appelant puisse utiliser la demande ultérieurement sans présenter ses propres informations d'identification.

Supposons par exemple qu'Alice ait accès à un objet Amazon Simple Storage Service (Amazon S3) et qu'elle souhaite partager temporairement l'accès à cet objet avec Bob. Alice peut générer une `GetObject` demande présignée à partager avec Bob afin qu'il puisse télécharger l'objet sans avoir besoin d'accéder aux informations d'identification d'Alice.

Principes de base de la présignature

Le SDK for Kotlin fournit des méthodes d'extension aux clients du service pour présigner les demandes. Toutes les demandes présignées nécessitent une durée qui représente la durée de validité de la demande signée. Une fois la durée terminée, la demande présignée expire et génère une erreur d'authentification si elle est exécutée.

Le code suivant montre un exemple qui crée une `GetObject` demande présignée pour Amazon S3. La demande est valide pendant 24 heures après sa création.

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = GetObjectRequest {
    bucket = "foo"
    key = "bar"
}

val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)
```

La méthode d'[presignGetObject](#) extension renvoie un [HttpRequest](#) objet. L'objet de demande contient le présigné URL où l'opération peut être invoquée. Un autre appelant peut utiliser la demande URL (ou l'intégralité de la demande) dans une base de code ou un environnement de langage de programmation différent.

Après avoir créé la demande présignée, utilisez un HTTP client pour appeler une demande. L'appel API à une HTTP GET demande dépend du HTTP client. L'exemple suivant utilise la [URL.readText](#) méthode Kotlin.

```
val objectContents = URL(presignedRequest.url.toString()).readText()
println(objectContents)
```

Configuration avancée de présignature

Dans le SDK, chaque méthode capable de présigner des demandes comporte une surcharge que vous pouvez utiliser pour fournir des options de configuration avancées, telles qu'une implémentation de signataire spécifique ou des paramètres de signature détaillés.

L'exemple suivant montre une `GetObject` demande Amazon S3 qui utilise la variante du CRT signataire et spécifie une future date de signature.

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = GetObjectRequest {
    bucket = "foo"
    key = "bar"
}

val presignedRequest = s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
    signingDate = Instant.now() + 24.hours
    expiresAfter = 8.hours
}
```

La demande présignée renvoyée est datée de 24 heures et n'est pas valide avant cette date. Il expire 8 heures plus tard.

Présignature POST et demandes PUT

De nombreuses opérations présignables ne nécessitent qu'un URL et doivent être exécutées sous forme HTTP GET de requêtes. Cependant, certaines opérations prennent un corps et doivent être exécutées sous forme de HTTP PUT requête HTTP POST ou avec des en-têtes dans certains cas. La présignature de ces demandes est identique à la présignature des GET demandes, mais l'invocation de la demande présignée est plus compliquée.

Voici un exemple de présignature d'une `PutObject` demande S3 :

```
val s3 = S3Client.fromEnvironment()
```

```

val unsignedRequest = PutObjectRequest {
    bucket = "foo"
    key = "bar"
}

val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

```

Le résultat renvoyé `HttpRequest` a une valeur de méthode de `HttpMethod.PUT` et inclut des en-têtes URL et qui doivent être inclus lors de la future invocation de la HTTP demande. Vous pouvez transmettre cette demande à un appelant qui pourra l'exécuter dans un autre environnement de base de code ou de langage de programmation.

Après avoir créé la PUT demande POST ou la présignée, utilisez un HTTP client pour appeler une demande. La API possibilité d'invoquer une PUT demande POST ou URL dépend du HTTP client utilisé. L'exemple suivant utilise un [OkHttp HTTPClient](#) et inclut un corps contenant `Hello world`.

```

val putRequest = Request
    .Builder()
    .url(presignedRequest.url.toString())
    .apply {
        presignedRequest.headers.forEach { key, values ->
            header(key, values.joinToString(", "))
        }
    }
    .put("Hello world".toRequestBody())
    .build()

val response = okHttp.newCall(putRequest).execute()

```

Opérations qu'ils SDK peuvent présigner

SDKfor Kotlin prend actuellement en charge la présignature API des opérations suivantes qui doivent être appelées avec la méthode associéeHTTP.

Service AWS	Opération	SDKméthode d'extension	HTTPMéthode d'utilisation
Amazon S3	GetObject	presignGetObject	HTTP GET
Amazon S3	PutObject	presignPutObject	HTTP PUT

Service AWS	Opération	SDK méthode d'extension	HTTP Méthode d'utilisation
Amazon S3	UploadPart	presignUploadPart	HTTP PUT
AWS Security Token Service	GetCallerIdentity	presignGetCallerId entité	HTTP POST
Amazon Polly	SynthesizeSpeech	presignSynthesizeSpeech	HTTP POST

Résolution des problèmes de FAQs

Lorsque vous l'utilisez Kit AWS SDK pour Kotlin dans vos applications, vous pouvez rencontrer certains des problèmes répertoriés dans cette rubrique. Utilisez les suggestions suivantes pour découvrir la cause première et résoudre l'erreur.

Comment résoudre les problèmes de « fermeture de la connexion » ?

Vous pouvez rencontrer des problèmes de « fermeture de connexion » en tant qu'exceptions telles que l'un des types suivants :

- `IOException: unexpected end of stream on <URL>`
- `EOFException: \n not found: limit=0`
- `HttpException: AWS_ERROR_HTTP_CONNECTION_CLOSED: The connection has closed or is closing.; crtErrorCode=2058; HttpStatusCode(CONNECTION_CLOSED)`

Ces exceptions indiquent qu'une TCP connexion entre le SDK et un service a été interrompue ou réinitialisée de façon inattendue. La connexion a peut-être été fermée par votre hébergeur, le AWS service ou un intermédiaire tel qu'une NAT passerelle, un proxy ou un équilibreur de charge.

Ces types d'exceptions sont automatiquement réessayés mais peuvent toujours apparaître dans les SDK journaux, en fonction de votre configuration de journalisation. Si l'exception est introduite dans votre code, cela indique que la stratégie de nouvelle tentative active a épuisé ses limites configurées, telles que le nombre maximum de tentatives ou le bucket de jetons de nouvelles tentatives. Consultez la [the section called “Nouvelle tentative”](#) section de ce guide pour plus d'informations sur les

stratégies de nouvelle tentative. Voir également [the section called “Pourquoi des exceptions sont-elles émises avant que le nombre maximum de tentatives n'ait été atteint ?”](#) le sujet ?

Pourquoi des exceptions sont-elles émises avant que le nombre maximum de tentatives n'ait été atteint ?

Parfois, vous pouvez voir des exceptions que vous vous attendiez à une nouvelle tentative, mais qui ont été rejetées à la place. Dans ces situations, les étapes suivantes peuvent vous aider à résoudre le problème.

- Vérifiez que l'exception peut être réessayée. Certaines exceptions ne sont pas réessayables, comme celles qui indiquent une demande de service mal formée, un manque d'autorisations ou l'inexistence de ressources, par exemple. Il SDK ne réessaie pas automatiquement ce type d'exceptions. Si vous détectez une exception qui hérite de `SdkBaseException`, vous pouvez vérifier la propriété booléenne `SdkBaseException.sdkErrorMetadata.isRetryable` pour vérifier si elle SDK a déterminé que l'exception peut être réessayée.
- Vérifiez que l'exception est introduite dans votre code. Certaines exceptions apparaissent dans les messages du journal sous forme d'informations mais ne sont pas réellement intégrées à votre code. Par exemple, les exceptions réessayables, telles que les erreurs de limitation, peuvent être enregistrées car elles fonctionnent SDK automatiquement sur plusieurs cycles. `backoff-and-retry` L'invocation d'une SDK opération génère une exception uniquement si elle n'a pas été gérée par les paramètres de nouvelle tentative configurés.
- Vérifiez les paramètres de nouvelle tentative que vous avez configurés. Consultez la [the section called “Nouvelle tentative”](#) section de ce guide pour plus d'informations sur les stratégies de nouvelle tentative et les politiques relatives aux nouvelles tentatives. Assurez-vous que votre code utilise les paramètres que vous attendez ou les valeurs par défaut automatiques.
- Pensez à ajuster vos paramètres de nouvelle tentative. Une fois que vous avez vérifié les éléments précédents, mais que le problème n'est pas résolu, vous pouvez envisager de modifier les paramètres de nouvelle tentative.
 - Augmentez le nombre maximum de tentatives. Par défaut, le nombre maximum de tentatives pour une opération est de 3. Si vous trouvez que cela ne suffit pas et que des exceptions se produisent toujours avec le paramètre par défaut, envisagez d'augmenter la `retryStrategy.maxAttempts` propriété dans la configuration de votre client. Pour plus d'informations, consultez [the section called “Nombre maximum de tentatives”](#).
 - Augmentez les paramètres de délai. Certaines exceptions peuvent être réessayées trop rapidement avant que la condition sous-jacente n'ait eu

l'occasion de se résoudre. Si vous pensez que c'est le cas, envisagez d'augmenter les `retryStrategy.delayProvider.maxBackoff` propriétés `retryStrategy.delayProvider.initialDelay` or dans la configuration de votre client. Pour plus d'informations, consultez [the section called "Retards et retards"](#).

- Désactive le mode disjoncteur. Par défaut, il SDK gère un bucket de jetons pour chaque client de service. Lorsque la demande est SDK tentée et qu'elle échoue avec une exception réessayable, le nombre de jetons est décrémenté ; lorsque la demande aboutit, le nombre de jetons est incrémenté.

Par défaut, si ce paquet de jetons atteint 0 jetons restants, le circuit est interrompu. Une fois le circuit interrompu, les nouvelles tentatives sont SDK désactivées et toutes les demandes en cours et suivantes qui échouent à la première tentative génèrent immédiatement une exception. Les nouvelles tentatives SDK de réactivation après des tentatives initiales réussies redonnent une capacité suffisante au bucket de jetons. Ce comportement est intentionnel et conçu pour éviter les tempêtes de nouvelles tentatives en cas de panne de service ou de rétablissement du service.

Si vous préférez SDK continuer à réessayer jusqu'au nombre maximum de tentatives configurées, envisagez de désactiver le mode disjoncteur en définissant la `retryStrategy.tokenBucket.useCircuitBreakerMode` propriété sur `false` dans la configuration de votre client. Lorsque cette propriété est définie sur `false`, le SDK client attend que le bucket de jetons atteigne une capacité suffisante plutôt que d'abandonner toute nouvelle tentative susceptible de provoquer une exception lorsqu'il ne reste aucun jeton.

Comment puis-je réparer **NoSuchMethodError** ou **NoClassDefFoundError** ?

Il s'agit d'un SDK appuie sur des dépendances diverses AWS et tierces pour fonctionner correctement. Si les dépendances attendues ne sont pas présentes au moment de l'exécution ou s'il s'agit d'une version inattendue, l'exception `NoSuchMethodError` d'exécution peut s'afficher.

Les conflits de dépendance se répartissent généralement en deux catégories : les conflits de dépendance SDK /Smithy et les conflits de dépendance liés à des tiers.

Lorsque vous créez une application Kotlin, vous gérez généralement les dépendances avec Gradle. L'ajout d'une dépendance à un client de SDK service à votre application devrait automatiquement résoudre et inclure toutes les dépendances transitives. Si votre application possède d'autres

dépendances, elles peuvent entrer en conflit avec les dépendances requises par le SDK (par exemple, OkHttp s'il s'agit d'un HTTP client couramment utilisé dont elles SDK dépendent).

Pour résoudre ces problèmes, vous devrez peut-être résoudre explicitement une version de dépendance spécifique ou des dépendances parallèles dans un espace de noms local afin d'éviter le conflit. La résolution des dépendances de Gradle est un sujet complexe qui est abordé dans les sections suivantes du manuel de l'utilisateur de Gradle.

- [Comprendre la résolution des dépendances](#)
- [Contraintes de dépendance et résolution des conflits](#)
- [Alignement des versions de dépendance](#)

SDK/Conflits de dépendance à Smithy

En général, les SDK modules 'dépendent d'autres SDK modules portant le même numéro de version. Par exemple, `aws.sdk.kotlin:s3:1.2.3` dépend de `aws.sdk.kotlin:aws-http:1.2.3`, qui dépend de `aws.sdk.kotlin:aws-core:1.2.3`, etc.

De plus, SDK les modules reposent également sur des versions de modules Smithy spécifiques et unifiées. Ces numéros de version de Smithy ne sont pas synchronisés avec les numéros de SDK version mais doivent toujours correspondre à la version attendue par le SDK. Par exemple, `aws.sdk.kotlin:s3:1.2.3` peut dépendre de `aws.smithy.kotlin:serde:1.1.1`, qui dépend de `aws.smithy.kotlin:runtime-core:1.1.1`, etc.

Si l'un de ces numéros de version ne correspond pas, vous risquez de rencontrer des conflits de dépendance. Assurez-vous de mettre à niveau toutes vos SDK dépendances à l'unisson et d'améliorer également toutes les dépendances explicites de Smithy à l'unisson. Envisagez d'utiliser notre [catalogue de versions Gradle](#) pour synchroniser les versions et éliminer les approximations entre les versions SDK et Smithy. Consultez la [the section called “Création de fichiers de construction de projets”](#) rubrique pour plus d'informations et d'exemples.

Sachez également que les modifications de version mineures dans les modules SDK /Smithy peuvent contenir des modifications importantes, comme décrit dans [la politique de gestion des versions SDK du](#) module. Lorsque vous effectuez une mise à niveau entre des versions mineures, veuillez particulièrement à examiner les journaux des modifications et à valider soigneusement le comportement d'exécution.

Je vois un `NoClassDefFoundError` pour `okhttp3.coroutines/ExecuteAsyncKt`

Si cette erreur s'affiche, cela signifie probablement que vous n'avez pas configuré votre client de service pour utiliser `OkHttp4Engine`. [Consultez la documentation](#) sur la façon de configurer Gradle et de l'utiliser `OkHttp4Engine` dans votre code.

Travaillez en Services AWS utilisant le Kit AWS SDK pour Kotlin

Ce chapitre contient des informations sur la façon de travailler avec Kotlin Services AWS en utilisant le SDK for Kotlin.

Table des matières

- [Travaillez avec Amazon S3 à l'aide du Kit AWS SDK pour Kotlin](#)
 - [Protection de l'intégrité des données avec des checksums](#)
 - [Charger un objet](#)
 - [Utiliser une valeur de somme de contrôle précalculée](#)
 - [Chargements partitionnés](#)
 - [Télécharger un objet](#)
 - [Validation asynchrone](#)
 - [Travaillez avec les points d'accès multirégionaux Amazon S3 en utilisant le SDK pour Kotlin](#)
 - [Travaillez avec des points d'accès multirégionaux](#)
 - [Travaillez avec des objets et des points d'accès multirégionaux](#)
- [Utilisez DynamoDB à l'aide du Kit AWS SDK pour Kotlin](#)
 - [Utiliser des points de AWS terminaison basés sur des comptes](#)
 - [Associez des classes à des éléments DynamoDB à l'aide du mappeur DynamoDB \(version préliminaire pour les développeurs\)](#)
 - [Commencez à utiliser DynamoDB Mapper](#)
 - [Ajouter des dépendances](#)
 - [Création et utilisation d'un mappeur](#)
 - [Définir un schéma avec des annotations de classe](#)
 - [Opérations d'appel](#)
 - [Travailler avec des réponses paginées](#)
 - [Configuration du mappeur DynamoDB](#)
 - [Utiliser des intercepteurs](#)
 - [Comprendre le pipeline de demandes](#)
 - [Hooks](#)

- [Hooks en lecture seule](#)
- [Modifier les crochets](#)
- [Ordre d'exécution](#)
- [Exemple de configuration](#)
- [Génération d'un schéma à partir d'annotations](#)
 - [Appliquer le plugin](#)
 - [Configurer le plugin](#)
 - [Annoter les classes](#)
 - [Annotations de classe](#)
 - [Annotations de propriétés](#)
 - [Définition d'un convertisseur d'articles personnalisé](#)
- [Définition manuelle de schémas](#)
 - [Définir un schéma dans le code](#)
- [Utiliser des index secondaires avec DynamoDB Mapper](#)
 - [Définition d'un schéma pour un index secondaire](#)
 - [Utiliser des indices secondaires dans les opérations](#)
- [Utiliser des expressions](#)
 - [Utiliser des expressions dans les opérations](#)
 - [Composants DSL](#)
 - [Attributs](#)
 - [Égalités et inégalités](#)
 - [Gammes et sets](#)
 - [Logique booléenne](#)
 - [Fonctions et propriétés](#)
 - [Trier les filtres clés](#)

Travaillez avec Amazon S3 à l'aide du Kit AWS SDK pour Kotlin

Votre interface principale avec Amazon Simple Storage Service pour le Kotlin SDK est le [S3Client](#). Utilisez les `S3Client` autres clients de service du SDK pour envoyer des [demandes](#) à Amazon S3.

Les ressources pour vous aider à utiliser le Kotlin SDK avec S3 sont les suivantes :

- la SDK [API référence Kotlin pour S3](#).
- le [guide de l'utilisateur du service S3](#) et la [API référence du service](#).

Les rubriques suivantes présentent des exemples de code guidé pour certains Kotlin SDK APIs compatibles avec S3.

Rubriques

- [Protection de l'intégrité des données avec des checksums](#)
- [Travaillez avec les points d'accès multirégionaux Amazon S3 en utilisant le SDK pour Kotlin](#)

Protection de l'intégrité des données avec des checksums

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 checksums avec Amazon S3, consultez le [guide de l'utilisateur d'Amazon Simple Storage Service](#), y compris les [algorithmes pris en charge](#).

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 fournir une valeur de somme de contrôle précalculée à l'aide de l'un des algorithmes pris en charge.

Note

À partir de la version 1.4.0 du Kit AWS SDK pour Kotlin, le SDK fournit des protections d'intégrité par défaut en calculant automatiquement une CRC32 somme de contrôle pour les téléchargements. Le SDK calcule cette somme de contrôle si vous ne fournissez pas de valeur de somme de contrôle précalculée ou si vous ne spécifiez pas d'algorithme que le SDK doit utiliser pour calculer une somme de contrôle.

Le SDK fournit également des paramètres globaux pour les protections de l'intégrité des données que vous pouvez définir en externe, que vous pouvez consulter dans le [Guide de référence AWS SDKs et sur les outils](#).

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

Vous chargez des objets sur Amazon S3 avec le SDK pour Kotlin en utilisant la [putObject](#) fonction avec un paramètre de requête. Le type de données de demande fournit la `checksumAlgorithm` propriété permettant le calcul de la somme de contrôle.

L'extrait de code suivant montre une demande de téléchargement d'un objet avec une CRC32 somme de contrôle. Lorsque le SDK envoie la demande, il calcule le CRC32 checksum et télécharge l'objet. Amazon S3 stocke le checksum avec l'objet.

```
val request = PutObjectRequest {  
    bucket = "amzn-s3-demo-bucket"  
    key = "key"  
    checksumAlgorithm = ChecksumAlgorithm.CRC32  
}
```

Si vous ne fournissez pas d'algorithme de somme de contrôle avec la demande, le comportement de somme de contrôle varie en fonction de la version du SDK que vous utilisez, comme indiqué dans le tableau suivant.

Comportement de somme de contrôle lorsqu'aucun algorithme de somme de contrôle n'est fourni

Version du SDK Kotlin	Comportement de Checksum
antérieur à la version 1.4.0	Le SDK ne calcule pas automatiquement une somme de contrôle basée sur le CRC et ne la fournit pas dans la demande.
1.4.0 ou version ultérieure	Le SDK utilise l' <code>CRC32</code> algorithme pour calculer le checksum et le fournit dans la demande. Amazon S3 valide l'intégrité du transfert en calculant sa propre somme de CRC32 contrôle et en la comparant à la somme de contrôle fournie par le SDK. Si les sommes de contrôle correspondent, la somme de contrôle est enregistrée avec l'objet.

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 SHA256 contrôle précalculée.

```
val request = PutObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    body = ByteStream.fromFile(File("file_to_upload.txt"))
    checksumAlgorithm = ChecksumAlgorithm.SHA256
    checksumSha256 = "cfb6d06da6e6f51c22ae3e549e33959dbb754db75a93665b8b579605464ce299"
}
```

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.

Vous devez spécifier l'algorithme de somme de contrôle dans la `CreateMultipartUpload` demande et dans chaque `UploadPart` demande. Enfin, vous devez spécifier la somme de contrôle de chaque partie du `CompleteMultipartUpload`. L'exemple suivant montre comment créer un téléchargement partitionné avec l'algorithme de somme de contrôle spécifié.

```
val multipartUpload = s3.createMultipartUpload {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumAlgorithm = ChecksumAlgorithm.Sha1
}

val partFilesToUpload = listOf("data-part1.csv", "data-part2.csv", "data-part3.csv")

val completedParts = partFilesToUpload
    .mapIndexed { i, fileName ->
        val uploadPartResponse = s3.uploadPart {
            bucket = "amzn-s3-demo-bucket"
            key = "key"
            body = ByteStream.fromFile(File(fileName))
            uploadId = multipartUpload.uploadId
        }
    }
```



```
        partNumber = i + 1 // Part numbers begin at 1.
        checksumAlgorithm = ChecksumAlgorithm.Sha1
    }

    CompletedPart {
        eTag = uploadPartResponse.eTag
        partNumber = i + 1
        checksumSha1 = uploadPartResponse.checksumSha1
    }
}

s3.completeMultipartUpload {
    uploadId = multipartUpload.uploadId
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    multipartUpload {
        parts = completedParts
    }
}
```

Télécharger un objet

Lorsque vous utilisez la méthode [GetObject](#) pour télécharger un objet, le SDK valide automatiquement le checksum la valeur de la clé est. `ChecksumMode.enabled` lorsque la `checksumMode` propriété du générateur pour le `GetObjectRequest` est définie sur `ChecksumMode.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.

```
val request = GetObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumMode = ChecksumMode.Enabled
}
```

Si l'objet n'a pas été chargé avec une somme de contrôle, aucune validation n'a lieu.

Si vous utilisez une version 1.4.0 ou ultérieure du SDK, le SDK vérifie automatiquement l'intégrité des `getObject` demandes sans ajouter de contenu `checksumMode = ChecksumMode.Enabled` à la demande.

Validation asynchrone

Comme le SDK pour Kotlin utilise des réponses en streaming lorsqu'il télécharge un objet depuis Amazon S3, la somme de contrôle sera calculée au fur et à mesure que vous consommez l'objet. Par conséquent, vous devez consommer l'objet pour que la somme de contrôle soit validée.

L'exemple suivant montre comment valider une somme de contrôle en consommant entièrement la réponse.

```
val request = GetObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumMode = checksumMode.Enabled
}

val response = s3Client.getObject(request) {
    println(response.body?.decodeToString()) // Fully consume the object.
    // The checksum is valid.
}
```

En revanche, le code de l'exemple suivant n'utilise pas l'objet de quelque manière que ce soit, de sorte que le checksum n'est pas validé.

```
s3Client.getObject(request) {
    println("Got the object.")
}
```

Si la somme de contrôle calculée par le SDK ne correspond pas à la somme de contrôle attendue envoyée avec la réponse, le SDK renvoie un `ChecksumMismatchException`

Travaillez avec les points d'accès multirégionaux Amazon S3 en utilisant le SDK pour Kotlin

Les points d'accès multirégionaux Amazon S3 fournissent un point de terminaison global que les applications peuvent utiliser pour traiter les demandes provenant de compartiments Amazon S3 situés dans plusieurs compartiments. Régions AWS Vous pouvez utiliser des points d'accès multirégionaux pour créer des applications multirégionales avec la même architecture que celle utilisée dans une seule région, puis exécuter ces applications n'importe où dans le monde.

Le guide de l'utilisateur Amazon S3 contient des informations générales supplémentaires sur les [points d'accès multirégionaux](#).

Travaillez avec des points d'accès multirégionaux

Pour créer un point d'accès multirégional, commencez par spécifier un compartiment dans chaque AWS région dans laquelle vous souhaitez traiter les demandes. L'extrait de code suivant crée deux compartiments.

Créer des compartiments

La fonction suivante crée deux compartiments destinés à fonctionner avec le point d'accès multirégional. Un compartiment se trouve dans la région us-east-1 et l'autre dans la région us-west-1.

La création du client S3 transmis en tant que premier argument est illustrée dans le premier exemple ci-dessous [the section called "Travailler avec des objets"](#).

```
suspend fun setUpTwoBuckets(
    s3: S3Client,
    bucketName1: String,
    bucketName2: String,
) {
    println("Create two buckets in different regions.")
    // The shared aws config file configures the default Region to be us-
east-1.
    s3.createBucket(
        CreateBucketRequest {
            bucket = bucketName1
        },
    )
    s3.waitUntilBucketExists {
        bucket = bucketName1
    }
    println("  Bucket [$bucketName1] created.")

    // Override the S3Client to work with us-west-1 for the second bucket.
    s3.withConfig {
        region = "us-west-1"
    }.use { s3West ->
        s3West.createBucket(
            CreateBucketRequest {
                bucket = bucketName2
                createBucketConfiguration = CreateBucketConfiguration {
                    locationConstraint = BucketLocationConstraint.UsWest1
                }
            }
        )
    }
}
```

```

        },
    )
    s3West.waitForBucketExists {
        bucket = bucketName2
    }
    println(" Bucket [$bucketName2] created.")
}
}

```

Vous utilisez le [client de contrôle S3 SDK](#) de Kotlin pour créer, supprimer et obtenir des informations sur les points d'accès multirégionaux.

Ajoutez une dépendance à l'artefact de contrôle S3, comme indiqué dans l'extrait suivant. (Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.)

```

...
implementation(platform("aws.sdk.kotlin:bom:X.Y.Z"))
implementation("aws.sdk.kotlin:s3control")
...

```

Configurez le client de contrôle S3 pour qu'il fonctionne Région AWS us-west-2 comme indiqué dans le code suivant. Toutes les opérations du client de contrôle S3 doivent cibler la us-west-2 région.

```

suspend fun createS3ControlClient(): S3ControlClient {
    // Configure your S3ControlClient to send requests to US West (Oregon).
    val s3Control = S3ControlClient.fromEnvironment {
        region = "us-west-2"
    }
    return s3Control
}

```

Utilisez le client de contrôle S3 pour créer un point d'accès multirégional en spécifiant les noms des compartiments (créés précédemment) comme indiqué dans le code suivant.

```

suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrapName: String,

```

```

): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrapName
                regions = listOf(
                    Region {
                        bucket = bucketName1
                    },
                    Region {
                        bucket = bucketName2
                    },
                )
            }
        }
    val requestToken: String? = createMrapResponse.requestTokenArn

    // Use the request token to check for the status of the
    CreateMultiRegionAccessPoint operation.
    if (requestToken != null) {
        waitForSucceededStatus(s3Control, requestToken, accountIdParam)
        println("MRAP created")
    }

    val getMrapResponse =
        s3Control.getMultiRegionAccessPoint(
            input = GetMultiRegionAccessPointRequest {
                accountId = accountIdParam
                name = mrapName
            },
        )
    val mrapAlias = getMrapResponse.accessPoint?.alias
    return "arn:aws:s3:::$accountIdParam:accesspoint/$mrpAlias"
}

```

La création d'un point d'accès multirégional étant une opération asynchrone, vous utilisez le jeton que vous recevez de la réponse immédiate pour vérifier l'état du processus de création. Une fois que la vérification de statut a renvoyé un message de réussite, vous pouvez utiliser l'GetMultiRegionAccessPointopération pour obtenir l'alias du point d'accès multirégional. L'alias est le dernier composant du ARN, dont vous avez besoin pour les opérations au niveau de l'objet.

Utiliser un jeton pour vérifier le statut

Utilisez le `DescribeMultiRegionAccessPointOperation` pour vérifier le statut de la dernière opération. Une fois que la `requestStatus` valeur devient `SUCCEEDED` « », vous pouvez utiliser le point d'accès multirégional.

```
suspend fun waitForSucceededStatus(
    s3Control: S3ControlClient,
    requestToken: String,
    accountIdParam: String,
    timeBetweenChecks: Duration = 1.minutes,
) {
    var describeResponse: DescribeMultiRegionAccessPointOperationResponse
    describeResponse = s3Control.describeMultiRegionAccessPointOperation(
        input = DescribeMultiRegionAccessPointOperationRequest {
            accountId = accountIdParam
            requestTokenArn = requestToken
        },
    )

    var status: String? = describeResponse.asyncOperation?.requestStatus
    while (status != "SUCCEEDED") {
        delay(timeBetweenChecks)
        describeResponse = s3Control.describeMultiRegionAccessPointOperation(
            input = DescribeMultiRegionAccessPointOperationRequest {
                accountId = accountIdParam
                requestTokenArn = requestToken
            },
        )
        status = describeResponse.asyncOperation?.requestStatus
        println(status)
    }
}
```

Travaillez avec des objets et des points d'accès multirégionaux

Vous utilisez le [client S3](#) pour travailler avec des objets dans des points d'accès multirégionaux. La plupart des opérations que vous effectuez sur des objets dans des compartiments peuvent être effectuées sur des points d'accès multirégionaux. Pour plus d'informations et une liste complète des opérations, consultez la section [Compatibilité des points d'accès multirégionaux avec les opérations S3](#).

Les opérations avec des points d'accès multirégionaux sont signées avec l'algorithme de signature asymétrique Sigv4 (Sigv4a). Support de SigV4a dans le monde Kit AWS SDK pour Kotlin actuel nécessite de signer avec le CRT signataire, ce qui constitue une dépendance distincte. Pour configurer la prise en charge de SigV4a, ajoutez les dépendances suivantes à votre projet. (Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.)

```
...
implementation(platform("aws.sdk.kotlin:bom:X.Y.Z"))
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))

implementation("aws.smithy.kotlin:aws-signing-crt")
implementation("aws.smithy.kotlin:http-auth-aws")
implementation("aws.sdk.kotlin:s3")
...
```

Après avoir ajouté les dépendances, configurez le client S3 pour qu'il utilise l'algorithme de signature Sigv4a comme indiqué dans le code suivant.

```
suspend fun createS3Client(): S3Client {
    // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a) signing
    algorithm.
    val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
    val s3 = S3Client.fromEnvironment {
        authSchemes = listOf(sigV4AScheme)
    }
    return s3
}
```

Après avoir configuré le client S3, les opérations prises en charge par S3 pour les points d'accès multirégionaux fonctionnent de la même manière. La seule différence est que le paramètre du compartiment doit être celui ARN du point d'accès multirégional. Vous pouvez les obtenir ARN depuis la console Amazon S3 ou par programmation, comme indiqué précédemment dans la `createMrap` fonction qui renvoie un. ARN

L'exemple de code suivant montre le code ARN utilisé dans une `GetObject` opération.

```
suspend fun getObjectFromMrap(
    s3: S3Client,
    mrapArn: String,
    keyName: String,
): String? {
```

```
    val request = GetObjectRequest {
        bucket = mrpArn // Use the ARN instead of the bucket name for object
operations.
        key = keyName
    }

    var stringObj: String? = null
    s3.getObject(request) { resp ->
        stringObj = resp.body?.decodeToString()
        if (stringObj != null) {
            println("Successfully read $keyName from $mrpArn")
        }
    }
    return stringObj
}
```

Utilisez DynamoDB à l'aide du Kit AWS SDK pour Kotlin

Utiliser des points de AWS terminaison basés sur des comptes

DynamoDB [AWS propose des points de terminaison basés sur des comptes](#) qui peuvent améliorer les performances en utilisant AWS votre identifiant de compte pour rationaliser le routage des demandes.

Pour profiter de cette fonctionnalité, vous devez utiliser la version 1.3.37 ou supérieure du Kit AWS SDK pour Kotlin Vous pouvez trouver la dernière version SDK répertoriée dans le [référentiel central de Maven](#). Lorsqu'une version prise en charge de SDK est active, elle utilise automatiquement les nouveaux points de terminaison.

Si vous souhaitez désactiver le routage basé sur le compte, quatre options s'offrent à vous :


- Configurez un client de service DynamoDB avec `AccountIdEndpointMode` le paramètre défini sur `DISABLED`
- Définissez une variable d'environnement.
- Définissez une propriété JVM système.
- Mettez à jour le paramètre du fichier de AWS configuration partagé.

L'extrait suivant illustre comment désactiver le routage basé sur un compte en configurant un client de service DynamoDB :


```
DynamoDbClient.fromEnvironment {  
    accountIdEndpointMode = AccountIdEndpointMode.DISABLED // The default value is  
    PREFERRED.  
}
```

Le guide de référence AWS SDKs and Tools fournit plus d'informations sur les [trois dernières options de configuration](#).

Associez des classes à des éléments DynamoDB à l'aide du mappeur DynamoDB (version préliminaire pour les développeurs)

 DynamoDB Mapper est une version préliminaire pour les développeurs. Les fonctionnalités ne sont pas complètes et sont susceptibles d'être modifiées.


[DynamoDB Mapper est une bibliothèque de haut niveau qui propose des mécanismes permettant de mapper les classes Kotlin aux tables et aux index DynamoDB, similaires au client DynamoDB amélioré ou au modèle de persistance des objets AWS SDK for Java de DynamoDB. AWS SDK for .NET](#)

Vous définissez des schémas qui décrivent votre objet de données et comment le convertir en éléments DynamoDB. Après avoir défini le schéma, DynamoDB Mapper fournit une interface intuitive permettant d'utiliser vos objets pour les opérations de création, de lecture, de mise à jour ou de suppression CRUD () sur vos tables et index.

Rubriques

- [Commencez à utiliser DynamoDB Mapper](#)
- [Configuration du mappeur DynamoDB](#)
- [Génération d'un schéma à partir d'annotations](#)
- [Définition manuelle de schémas](#)
- [Utiliser des index secondaires avec DynamoDB Mapper](#)
- [Utiliser des expressions](#)

Commencez à utiliser DynamoDB Mapper

 DynamoDB Mapper est une version préliminaire pour les développeurs. Les fonctionnalités ne sont pas complètes et sont susceptibles d'être modifiées.

Le didacticiel suivant présente les composants de base de DynamoDB Mapper et montre comment les utiliser dans votre code.

Ajouter des dépendances

Pour commencer à utiliser DynamoDB Mapper dans votre projet Gradle, ajoutez un plugin et deux dépendances à votre fichier `build.gradle.kts`

(Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.)

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

plugins {
    id("aws.sdk.kotlin.hll.dynamodbmapper.schema.generator") version "$sdkVersion-beta" // For the Developer Preview, use the beta version of the latest SDK.
}

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta")
    implementation("aws.sdk.kotlin:dynamodb-mapper-annotations:$sdkVersion-beta")
}
```

*Remplacez `<Version>` par la dernière version du SDK. Pour trouver la dernière version du SDK, consultez la [dernière version sur GitHub](#).

Note

Certaines de ces dépendances sont facultatives si vous prévoyez de définir des schémas manuellement. Consultez [the section called “Définition manuelle de schémas”](#) pour plus d'informations et pour découvrir l'ensemble réduit de dépendances.

Création et utilisation d'un mappeur

Le mappeur DynamoDB utilise le client DynamoDB pour Kit AWS SDK pour Kotlin interagir avec DynamoDB. Vous devez fournir une instance entièrement configurée lorsque vous créez une [DynamoDbClient](#) instance de mappeur, comme indiqué dans l'extrait de code suivant :

```
import aws.sdk.kotlin.hll.dynamodbmapper.DynamoDbMapper
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient

val client = DynamoDbClient.fromEnvironment()
val mapper = DynamoDbMapper(client)
```

Après avoir créé l'instance du mappeur, vous pouvez l'utiliser pour obtenir l'instance de table comme indiqué ci-dessous :

```
val carsTable = mapper.getTable("cars", CarSchema)
```

Le code précédent obtient une référence à une table DynamoDB nommée `cars` avec un schéma défini par `CarSchema` (nous discutons des schémas ci-dessous). Après avoir créé une instance de table, vous pouvez effectuer des opérations sur celle-ci. L'extrait de code suivant montre deux exemples d'opérations sur la `cars` table :

```
carsTable.putItem {
    item = Car(make = "Ford", model = "Model T", ...)
}

carsTable
    .queryPaginated {
        keyCondition = KeyFilter(partitionKey = "Peugeot")
    }
    .items()
    .collect { car -> println(car) }
```

Le code précédent crée un nouvel élément dans le `cars` tableau. Le code crée une `Car` instance en ligne à l'aide de la `Car` classe, dont la définition est présentée ci-dessous. Ensuite, le code interroge la `cars` table pour les éléments dont la clé de partition est la clé `Peugeot` et les imprime. Les opérations sont [décrites plus en détail ci-dessous](#).

Définir un schéma avec des annotations de classe

Pour diverses classes Kotlin, le SDK peut générer automatiquement des schémas au moment de la construction à l'aide du plugin DynamoDB Mapper Schema Generator pour Gradle. Lorsque vous utilisez le générateur de schéma, le SDK inspecte vos classes pour en déduire le schéma, ce qui simplifie en partie la complexité liée à la définition manuelle des schémas. Vous pouvez personnaliser le schéma généré à l'aide d'[annotations](#) et de [configurations](#) supplémentaires.

Pour générer un schéma à partir d'annotations, commencez par annoter vos classes avec [@DynamoDbItem](#) et toutes les touches. [@DynamoDbPartitionKey](#) [@DynamoDbSortKey](#) Le code suivant montre la Car classe annotée :

```
// The annotations used in the Car class are used by the plugin to generate a schema.
@DynamoDbItem
data class Car(
    @DynamoDbPartitionKey
    val make: String,

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

Après la construction, vous pouvez vous référer à la génération automatique `CarSchema`. Vous pouvez utiliser la référence dans la `getTable` méthode du mappeur pour obtenir une instance de table, comme indiqué ci-dessous :

```
import aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas.CarSchema

// `CarSchema` is generated at build time.
val carsTable = mapper.getTable("cars", CarSchema)
```

Vous pouvez également obtenir l'instance de table en tirant parti d'une méthode d'extension générée automatiquement au moment de la construction. [DynamoDbMapper](#) En utilisant cette approche, vous n'avez pas besoin de faire référence au schéma par son nom. Comme indiqué ci-dessous, la méthode d'`getCarsTable` extension générée automatiquement renvoie une référence à l'instance de table :

```
val carsTable = mapper.getCarsTable("cars")
```

Pour obtenir des détails et des exemples, consultez [the section called “Génération d'un schéma”](#).

Opérations d'appel

DynamoDB Mapper prend en charge un sous-ensemble des opérations disponibles sur les SDK. `DynamoDbClient` Les opérations de mappage portent le même nom que leurs homologues sur le client SDK. De nombreux membres de demande/réponse du mappeur sont les mêmes que leurs homologues clients du SDK, bien que certains aient été renommés, retapés ou complètement supprimés.

Vous invoquez une opération sur une instance de table à l'aide d'une syntaxe DSL, comme illustré ci-dessous :

```
import aws.sdk.kotlin.h11.dynamodbmapper.operations.putItem
import aws.sdk.kotlin.services.dynamodb.model.ReturnConsumedCapacity

val putResponse = carsTable.putItem {
    item = Car(make = "Ford", model = "Model T", ...)
    returnConsumedCapacity = ReturnConsumedCapacity.Total
}

println(putResponse.consumedCapacity)
```

Vous pouvez également invoquer une opération en utilisant un objet de requête explicite :

```
import aws.sdk.kotlin.h11.dynamodbmapper.operations.PutItemRequest
import aws.sdk.kotlin.services.dynamodb.model.ReturnConsumedCapacity

val putRequest = PutItemRequest<Car> {
    item = Car(make = "Ford", model = "Model T", ...)
    returnConsumedCapacity = ReturnConsumedCapacity.Total
}

val putResponse = carsTable.putItem(putRequest)
println(putResponse.consumedCapacity)
```

Les deux exemples de code précédents sont équivalents.

Travailler avec des réponses paginées

Certaines opérations, comme `query` et `scan` peuvent renvoyer des ensembles de données qui peuvent être trop volumineux pour être renvoyés en une seule réponse. Pour garantir le traitement

de tous les objets, DynamoDB Mapper fournit des méthodes de pagination qui n'appellent pas DynamoDB immédiatement, mais renvoient [Flow](#) une réponse du type de réponse à l'opération, comme illustré ci-dessous : `Flow<ScanResponse<Car>>`

```
import aws.sdk.kotlin.h11.dynamodbmapper.operations.scanPaginated

val scanResponseFlow = carsTable.scanPaginated { }

scanResponseFlow.collect { response ->
    val items = response.items.orEmpty()
    println("Found page with ${items.size} items:")

    items.forEach { car -> println(car) }
}
```


Souvent, un flux d'objets est plus utile à la logique métier qu'un flux de réponses contenant des objets. Le mappeur fournit une méthode d'extension des réponses paginées pour accéder au flux d'objets. Par exemple, le code suivant renvoie un `Flow<Car>` plutôt qu'un `Flow<ScanResponse<Car>>` comme indiqué précédemment :

```
import aws.sdk.kotlin.h11.dynamodbmapper.operations.items
import aws.sdk.kotlin.h11.dynamodbmapper.operations.scanPaginated

val carFlow = carsTable
    .scanPaginated { }
    .items()

carFlow.collect { car -> println(car) }
```

Configuration du mappeur DynamoDB

 DynamoDB Mapper est une version préliminaire pour les développeurs. Les fonctionnalités ne sont pas complètes et sont susceptibles d'être modifiées.

DynamoDB Mapper propose des options de configuration que vous pouvez utiliser pour personnaliser le comportement de la bibliothèque en fonction de votre application.

Utiliser des intercepteurs

La bibliothèque DynamoDB Mapper définit les hooks que vous pouvez exploiter aux étapes critiques du pipeline de requêtes du mappeur. Vous pouvez implémenter l'[Interceptor](#) interface pour implémenter des hooks afin d'observer ou de modifier le processus du mappeur.

Vous pouvez enregistrer un ou plusieurs intercepteurs sur un seul mappeur DynamoDB en tant qu'option de configuration. Consultez l'[exemple](#) à la fin de cette section pour savoir comment enregistrer un intercepteur.

Comprendre le pipeline de demandes

Le pipeline de requêtes du mappeur comprend les 5 étapes suivantes :

1. Initialisation : configurez l'opération et collectez le contexte initial.
2. Sérialisation : convertissez les objets de demande de haut niveau en objets de demande de bas niveau. Cette étape convertit les objets Kotlin de haut niveau en éléments DynamoDB composés de noms et de valeurs d'attributs.
3. Invocation de bas niveau : exécutez une demande sur le client DynamoDB sous-jacent.
4. Désérialisation : convertissez les objets de réponse de bas niveau en objets de réponse de haut niveau. Cette étape inclut la conversion d'éléments DynamoDB composés de noms et de valeurs d'attributs en objets Kotlin de haut niveau.
5. Achèvement : finalisez la réponse de haut niveau à renvoyer à l'appelant. Si une exception a été émise lors de l'exécution du pipeline, cette étape finalise l'exception envoyée à l'appelant.

Hooks

Les hooks sont des méthodes d'interception que le mappeur invoque avant ou après des étapes spécifiques du pipeline. Il existe deux variantes de hooks : lecture seule et modification (ou lecture-écriture). Par exemple, `readBeforeInvocation` il s'agit d'un hook en lecture seule que le mappeur exécute dans la phase précédant l'étape d'invocation de bas niveau.

Hooks en lecture seule

Le mappeur invoque des hooks en lecture seule avant et après chaque étape du pipeline (sauf avant l'étape d'initialisation et après l'étape d'achèvement). Les capots en lecture seule offrent une vue en lecture seule d'une opération de haut niveau en cours. Ils fournissent un mécanisme permettant d'examiner l'état d'une opération pour la journalisation, le débogage, la collecte de métriques, par exemple. Chaque hook en lecture seule reçoit un argument de contexte et le renvoie. [Unit](#)

Le mappeur détecte toute exception émise lors d'un hook en lecture seule et l'ajoute au contexte. Il transmet ensuite le contexte à l'exception des hooks d'interception suivants au cours de la même phase. Le mappeur envoie une exception à l'appelant uniquement après avoir appelé le hook en lecture seule du dernier intercepteur pour la même phase. Par exemple, si un mappeur est configuré avec deux intercepteurs A et B que le `readAfterSerialization` crochet génère une exception, le mappeur ajoute l'exception au contexte transmis au B crochet. A `readAfterSerialization` Une fois B le `readAfterSerialization` hook terminé, le mappeur renvoie l'exception à l'appelant.

Modifier les crochets

Le mappeur invoque les hooks de modification avant chaque étape du pipeline (sauf avant l'initialisation). Les hooks de modification offrent la possibilité de voir et de modifier une opération de haut niveau en cours. Ils peuvent être utilisés pour personnaliser le comportement et les données, contrairement à la configuration du mappeur et aux schémas d'éléments. Chaque crochet de modification reçoit un argument de contexte et renvoie un sous-ensemble de ce contexte en conséquence, soit modifié par le crochet, soit transmis depuis le contexte d'entrée.

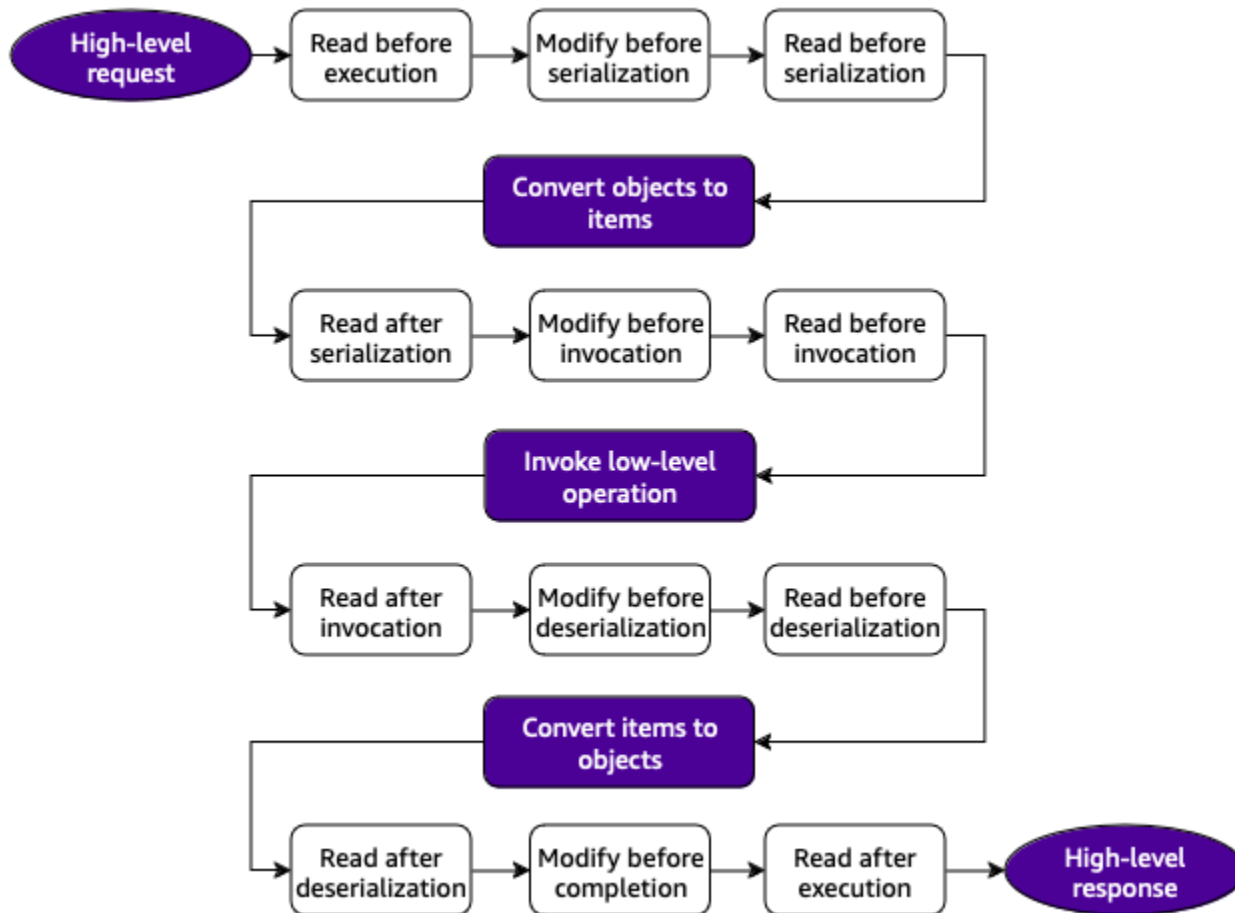
Si le mappeur détecte une exception pendant qu'il exécute un hook de modification, il n'exécute aucun hook de modification d'un autre intercepteur au cours de la même phase. Le mappeur ajoute l'exception au contexte et la transmet au hook en lecture seule suivant. Le mappeur envoie une exception à l'appelant uniquement après avoir appelé le dernier hook en lecture seule des intercepteurs pour la même phase. Par exemple, si un mappeur est configuré avec deux intercepteurs A et B que A son `modifyBeforeSerialization` crochet génère une exception, B le `modifyBeforeSerialization` crochet ne sera pas invoqué. Les interceptors A et B ' s `readAfterSerialization` hook s'exécuteront, après quoi l'exception sera renvoyée à l'appelant.

Ordre d'exécution

L'ordre dans lequel les intercepteurs sont définis dans la configuration d'un mappeur détermine l'ordre dans lequel le mappeur appelle les hooks :

- Pour les phases précédant l'étape d'invocation de bas niveau, il exécute les hooks dans le même ordre que celui dans lequel ils ont été ajoutés dans la configuration.
- Pour les phases suivant l'étape d'invocation de bas niveau, il exécute les hooks dans l'ordre inverse de celui dans lequel ils ont été ajoutés dans la configuration.

Le schéma suivant montre l'ordre d'exécution des méthodes hook :



Description textuelle de l'ordre d'exécution des méthodes hook

Un mappeur exécute les hooks d'un intercepteur dans l'ordre suivant :

1. Le mappeur DynamoDB invoque une requête de haut niveau
2. À lire avant l'exécution
3. Modifier avant la sérialisation
4. À lire avant la sérialisation
5. Le mappeur DynamoDB convertit les objets en éléments
6. Lire après la sérialisation
7. Modifier avant l'invocation
8. À lire avant l'invocation
9. Le mappeur DynamoDB invoque l'opération de bas niveau
10. À lire après l'invocation
11. Modifier avant la désérialisation

12À lire avant la désérialisation

13Le mappeur DynamoDB convertit les éléments en objets

14Lire après désérialisation

15Modifier avant de terminer

16Lire après l'exécution

17Le mappeur DynamoDB renvoie une réponse de haut niveau

Exemple de configuration

L'exemple suivant montre comment configurer un intercepteur sur une DynamoDbMapper instance :


```
import aws.sdk.kotlin.h11.dynamodbmapper.DynamoDbMapper
import aws.sdk.kotlin.h11.dynamodbmapper.operations.ScanRequest
import aws.sdk.kotlin.h11.dynamodbmapper.operations.ScanResponse
import aws.sdk.kotlin.h11.dynamodbmapper.pipeline.Interceptor
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.sdk.kotlin.services.dynamodb.model.ScanRequest as LowLevelScanRequest
import aws.sdk.kotlin.services.dynamodb.model.ScanResponse as LowLevelScanResponse

val printingInterceptor = object : Interceptor<User, ScanRequest<User>,
    LowLevelScanRequest, LowLevelScanResponse, ScanResponse<User>> {
    override fun readBeforeDeserialization(ctx: LResContext<User, ScanRequest<User>,
        LowLevelScanRequest, LowLevelScanResponse>) {
        println("Scan response contains ${ctx.lowLevelResponse.count} items.")
    }
}

val client = DynamoDbClient.fromEnvironment()

val mapper = DynamoDbMapper(client) {
    interceptors += printingInterceptor
}
```

Génération d'un schéma à partir d'annotations

 DynamoDB Mapper est une version préliminaire pour les développeurs. Les fonctionnalités ne sont pas complètes et sont susceptibles d'être modifiées.

Le mappeur DynamoDB repose sur des schémas qui définissent le mappage entre vos classes Kotlin et les éléments DynamoDB. Vos classes Kotlin peuvent piloter la création de schémas en utilisant le plugin Gradle générateur de schémas.

Appliquer le plugin

Pour commencer à générer des schémas de code pour vos classes, appliquez le plugin dans le script de compilation de votre application et ajoutez une dépendance au module d'annotations. L'extrait de script Gradle suivant montre la configuration nécessaire à la génération de code.

(Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.)

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

plugins {
    id("aws.sdk.kotlin.hll.dynamodbmapper.schema.generator") version "$sdkVersion-beta" // For the Developer Preview, use the beta version of the latest SDK.
}

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta")
    implementation("aws.sdk.kotlin:dynamodb-mapper-annotations:$sdkVersion-beta")
}
```

Configurer le plugin

Le plugin propose un certain nombre d'options de configuration que vous pouvez appliquer en utilisant l'extension du `dynamoDbMapper { ... }` plugin dans votre script de compilation :

Option	Description de l'option	Valeurs
<code>generateBuilderClasses</code>	Contrôle si les classes de générateur de DSL style - style seront générées pour les classes annotées avec <code>@DynamoDbItem</code>	<code>WHEN_REQUIRED</code> (par défaut) : les classes Builder ne seront pas générées pour les classes composées uniquement de membres mutables publics et ayant un constructeur zéro-arg

Option	Description de l'option	Valeurs
		ALWAYS: les classes Builder seront toujours générées
<code>visibility</code>	Contrôle la visibilité des classes générées	PUBLIC (default) INTERNAL
<code>destinationPackage</code>	Spécifie le nom du package pour les classes générées	RELATIVE(par défaut) : les classes de schéma seront générées dans un sous-package relatif à votre classe annotée. Par défaut, le sous-package est nommé <code>dynamodbMapper.generatedschemas</code> , et cela est configurable en passant un paramètre de chaîne ABSOLUTE: Les classes de schéma seront générées dans un package absolu relatif à la racine de votre application. Par défaut, le package est nommé <code>aws.sdk.kotlin.hll.dynamodbMapper.generatedschemas</code> , et cela est configurable en passant un paramètre de chaîne.
<code>generateGetTableExtension</code>	Contrôle si une méthode d' <code>DynamoDbMapper.getTableExtension</code> sera générée	true (default) false

Exemple Exemple de configuration d'un plugin de génération de code

L'exemple suivant configure le package de destination et la visibilité du schéma généré :

```
// build.gradle.kts

import aws.sdk.kotlin.h11.dynamodbmapper.codegen.annotations.DestinationPackage
import aws.sdk.kotlin.h11.dynamodbmapper.codegen.annotations.Visibility
import aws.smithy.kotlin.runtime.ExperimentalApi

@OptIn(ExperimentalApi::class)
dynamoDbMapper {
    destinationPackage = DestinationPackage.RELATIVE("my.configured.package")
    visibility = Visibility.INTERNAL
}
```

Annoter les classes

Le générateur de schéma recherche des annotations de classe afin de déterminer les classes pour lesquelles générer des schémas. Pour choisir de générer des schémas, annotez vos classes avec [@DynamoDbItem](#). Vous devez également annoter une propriété de classe qui sert de clé de partition à l'élément avec l'[@DynamoDbPartitionKey](#) annotation.

La définition de classe suivante indique les annotations minimales requises pour la génération du schéma :

Exemple

```
@DynamoDbItem
data class Employee(
    @DynamoDbPartitionKey
    val id: Int,

    val name: String,
    val role: String,
)
```

Annotations de classe

Les annotations suivantes sont appliquées aux classes pour contrôler la génération de schémas :

- `@DynamoDbItem`: Spécifie que cette classe/interface décrit un type d'élément dans une table. Toutes les propriétés publiques de ce type seront mappées à des attributs à moins qu'elles ne soient explicitement ignorées. Lorsqu'il est présent, un schéma sera généré pour cette classe.
- `converterName`: paramètre facultatif qui indique qu'un schéma personnalisé doit être utilisé plutôt que celui créé par le plugin générateur de schémas. Il s'agit du nom complet de la `ItemConverter` classe personnalisée. La [the section called “Définition d'un convertisseur d'articles personnalisé”](#) section présente un exemple de création et d'utilisation d'un schéma personnalisé.

Annotations de propriétés

Vous pouvez appliquer les annotations suivantes aux propriétés de classe pour contrôler la génération du schéma :

- [@DynamoDbPartitionKey](#): Spécifie la clé de partition de l'élément.
- [@DynamoDbSortKey](#): Spécifie une clé de tri facultative pour l'élément.
- [@DynamoDbIgnore](#): Spécifie que cette propriété de classe ne doit pas être convertie vers/depuis un attribut Item par le mappeur DynamoDB.
- [@DynamoDbAttribute](#): Spécifie un nom d'attribut personnalisé facultatif pour cette propriété de classe.

Définition d'un convertisseur d'articles personnalisé

Dans certains cas, vous souhaitez peut-être définir un convertisseur d'éléments personnalisé pour votre classe. Cela s'explique notamment par le fait que votre classe utilise un type qui n'est pas pris en charge par le plugin du générateur de schéma. Nous utilisons la version suivante de la `Employee` classe comme exemple :

```
import kotlin.uuid.Uuid

@DynamoDbItem
data class Employee(
    @DynamoDbPartitionKey
    var id: Int,

    var name: String,
    var role: String,
    var workstationId: Uuid
```

)

La `Employee` classe utilise désormais un `kotlin.uuid.Uuid` type qui n'est actuellement pas pris en charge par le générateur de schéma. La génération du schéma échoue avec une erreur `:Unsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)`. Cette erreur indique que le plugin ne peut pas générer de convertisseur d'éléments pour cette classe. Par conséquent, nous devons écrire les nôtres.

Pour ce faire, nous implémentons un [ItemConverter](#) pour la classe, puis modifions l'annotation de `@DynamoDbItem` classe en spécifiant le nom complet du nouveau convertisseur d'éléments.

Tout d'abord, nous implémentons un [ValueConverter](#) pour la `kotlin.uuid.Uuid` classe :

```
import aws.sdk.kotlin.h11.dynamodbmapper.values.ValueConverter
import aws.sdk.kotlin.services.dynamodb.model.AttributeValue
import kotlin.uuid.Uuid

public val UuidValueConverter = object : ValueConverter<Uuid> {
    override fun convertFrom(to: AttributeValue): Uuid =
        Uuid.parseHex(to.asS())

    override fun convertTo(from: Uuid): AttributeValue =
        AttributeValue.S(from.toHexString())
}
```

Ensuite, nous implémentons un `ItemConverter` pour notre `Employee` classe. Il `ItemConverter` utilise ce nouveau convertisseur de valeurs dans le descripteur d'attribut pour `workstationId` « » :

```
import aws.sdk.kotlin.h11.dynamodbmapper.items.AttributeDescriptor
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.h11.dynamodbmapper.items.SimpleItemConverter
import aws.sdk.kotlin.h11.dynamodbmapper.values.scalars.IntConverter
import aws.sdk.kotlin.h11.dynamodbmapper.values.scalars.StringConverter

public object MyEmployeeConverter : ItemConverter<Employee> by SimpleItemConverter(
    builderFactory = { Employee() },
    build = { this },
    descriptors = arrayOf(
        AttributeDescriptor(
            "id",
```

```

        Employee::id,
        Employee::id::set,
        IntConverter,
    ),
    AttributeDescriptor(
        "name",
        Employee::name,
        Employee::name::set,
        StringConverter,
    ),
    AttributeDescriptor(
        "role",
        Employee::role,
        Employee::role::set,
        StringConverter
    ),
    AttributeDescriptor(
        "workstationId",
        Employee::workstationId,
        Employee::workstationId::set,
        UuidValueConverter
    )
),
)
)

```

Maintenant que nous avons défini le convertisseur d'articles, nous pouvons l'appliquer à notre classe. Nous mettons à jour l'[@DynamoDbItem](#) annotation pour faire référence au convertisseur d'articles en fournissant le nom de classe complet, comme indiqué ci-dessous :

```

import kotlin.uuid.Uuid

@DynamoDbItem("my.custom.item.converter.MyEmployeeConverter")
data class Employee(
    @DynamoDbPartitionKey
    var id: Int,

    var name: String,
    var role: String,
    var workstationId: Uuid
)

```

Enfin, nous pouvons commencer à utiliser la classe avec DynamoDB Mapper.

Définition manuelle de schémas

⚠ DynamoDB Mapper est une version préliminaire pour les développeurs. Les fonctionnalités ne sont pas complètes et sont susceptibles d'être modifiées.

Définir un schéma dans le code

Pour un contrôle et une personnalisation optimaux, vous pouvez définir et personnaliser manuellement les schémas dans le code.

Comme indiqué dans l'extrait suivant, vous devez inclure moins de dépendances dans votre `build.gradle.kts` fichier par rapport à la création de schéma pilotée par des annotations.

(Vous pouvez accéder au [X.Y.Z](#) lien pour voir la dernière version disponible.)

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta") // For the
    Developer Preview, use the beta version of the latest SDK.
}
```

Notez que vous n'avez pas besoin du plugin générateur de schéma ni du package d'annotations.

Le mappage entre une classe Kotlin et un élément DynamoDB nécessite [ItemSchema<T>](#) une implémentation, T où est le type de la classe Kotlin. Un schéma comprend les éléments suivants :

- Un convertisseur d'éléments, qui définit comment convertir entre les instances d'objets Kotlin et les éléments DynamoDB.
- Spécification de clé de partition, qui définit le nom et le type de l'attribut de clé de partition.
- Facultativement, une spécification de clé de tri, qui définit le nom et le type de l'attribut de clé de tri.

Dans le code suivant, nous créons manuellement une `CarSchema` instance :

```
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemSchema
```

```
import aws.sdk.kotlin.h11.dynamodbmapper.model.itemOf

// We define a schema for this data class.
data class Car(val make: String, val model: String, val initialYear: Int)

// First, define an item converter.
val carConverter = object : ItemConverter<Car> {
    override fun convertTo(from: Car, onlyAttributes: Set<String>?): Item = itemOf(
        "make" to AttributeValue.S(from.make),
        "model" to AttributeValue.S(from.model),
        "initialYear" to AttributeValue.N(from.initialYear.toString()),
    )

    override fun convertFrom(to: Item): Car = Car(
        make = to["make"]?.asSOrNull() ?: error("Invalid attribute `make`"),
        model = to["model"]?.asSOrNull() ?: error("Invalid attribute `model`"),
        initialYear = to["initialYear"]?.asNOrNull()?.toIntOrNull()
            ?: error("Invalid attribute `initialYear`"),
    )
}

// Next, define the specifications for the partition key and sort key.
val makeKey = KeySpec.String("make")
val modelKey = KeySpec.String("model")

// Finally, create the schema from the converter and key specifications.
// Note that the KeySpec for the partition key comes first in the ItemSchema
// constructor.
val CarSchema = ItemSchema(carConverter, makeKey, modelKey)
```

Le code précédent crée un convertisseur nommé `carConverter`, qui est défini comme une implémentation anonyme de `ItemConverter<Car>`. La `convertTo` méthode du convertisseur accepte un `Car` argument et renvoie une `Item` instance représentant les clés littérales et les valeurs des attributs des éléments DynamoDB. La `convertFrom` méthode du convertisseur accepte un `Item` argument et renvoie une `Car` instance à partir des valeurs d'attribut de l'`Item` argument.

Le code crée ensuite deux spécifications clés : une pour la clé de partition et une pour la clé de tri. Chaque table ou index DynamoDB doit avoir exactement une clé de partition, de même que chaque définition de schéma DynamoDB Mapper. Les schémas peuvent également avoir une clé de tri.

Dans la dernière instruction, le code crée un schéma pour la table `cars` DynamoDB à partir du convertisseur et des spécifications clés.

Le schéma obtenu est équivalent au schéma piloté par les annotations que nous avons généré dans la [the section called “Définir un schéma avec des annotations de classe”](#) section. À titre de référence, voici la classe annotée que nous avons utilisée :

Classe de véhicule avec annotations DynamoDB Mapper

```
@DynamoDbItem
data class Car(
    @DynamoDbPartitionKey
    val make: String,


    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

Outre l'implémentation de la `votreItemConverter`, DynamoDB Mapper inclut plusieurs implémentations utiles, telles que :

- [SimpleItemConverter](#): fournit une logique de conversion simple en utilisant une classe de générateur et des descripteurs d'attributs. Consultez l'exemple ci-dessous [the section called “Définition d'un convertisseur d'articles personnalisé”](#) pour savoir comment utiliser cette implémentation.
- [HeterogeneousItemConverter](#): fournit une logique de conversion de type polymorphe en utilisant un attribut discriminatoire et en `ItemConverter` déléguant des instances pour les sous-types.
- [DocumentConverter](#): fournit une logique de conversion pour les données non structurées contenues dans des [Document](#) objets.

Utiliser des index secondaires avec DynamoDB Mapper

 DynamoDB Mapper est une version préliminaire pour les développeurs. Les fonctionnalités ne sont pas complètes et sont susceptibles d'être modifiées.

Définition d'un schéma pour un index secondaire

Les tables DynamoDB prennent en charge des index secondaires qui permettent d'accéder aux données à l'aide de clés différentes de celles définies dans la table de base elle-même. Comme pour les tables de base, DynamoDB Mapper interagit avec les indices en utilisant le type. [ItemSchema](#)

Les index secondaires DynamoDB ne doivent pas nécessairement contenir tous les attributs de la table de base. Par conséquent, la classe Kotlin qui correspond à un index peut être différente de la classe Kotlin qui correspond à la table de base de cet index. Dans ce cas, un schéma distinct doit être déclaré pour la classe d'index.

Le code suivant crée manuellement un schéma d'index pour la table cars DynamoDB.

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.hll.dynamodbmapper.model.itemOf

// This is a data class for modelling the index of the Car table. Note
// that it contains a subset of the fields from the Car class and also
// uses different names for them.
data class Model(val name: String, val manufacturer: String)

// We define an item converter.
val modelConverter = object : ItemConverter<Model> {
    override fun convertTo(from: Model, onlyAttributes: Set<String>?): Item = itemOf(
        "model" to AttributeValue.S(from.name),
        "make" to AttributeValue.S(from.manufacturer),
    )

    override fun convertFrom(to: Item): Model = Model(
        name = to["model"]?.asSOrNull() ?: error("Invalid attribute `model`"),
        manufacturer = to["make"]?.asSOrNull() ?: error("Invalid attribute `make`"),
    )
}
val modelKey = KeySpec.String("model")
val makeKey = KeySpec.String("make")

val modelSchema = ItemSchema(modelConverter, modelKey, makeKey) // The partition key
specification is the second parameter.

/* Note that `Model` index's partition key is `model` and its sort key is `make`,
    whereas the `Car` base table uses `make` as the partition key and `model` as the
    sort key:
```

```
@DynamoDbItem
data class Car(
    @DynamoDbPartitionKey
    val make: String,

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
*/
```

Nous pouvons désormais utiliser des `Model` instances dans les opérations.

Utiliser des indices secondaires dans les opérations


Le mappeur DynamoDB prend en charge un sous-ensemble d'opérations sur les indices, à savoir `et` `queryPaginated` `scanPaginated`. Pour appeler ces opérations sur un index, vous devez d'abord obtenir une référence à un index à partir de l'objet de table. Dans l'exemple suivant, nous utilisons celui `modelSchema` que nous avons créé précédemment pour `'cars-by-model'` (création non illustrée ici) :

```
val table = mapper.getTable("cars", CarSchema)
val index = table.getIndex("cars-by-model", modelSchema)

val modelFlow = index
    .scanPaginated { }
    .items()

modelFlow.collect { model -> println(model) }
```

Utiliser des expressions

 DynamoDB Mapper est une version préliminaire pour les développeurs. Les fonctionnalités ne sont pas complètes et sont susceptibles d'être modifiées.

Certaines opérations DynamoDB [acceptent](#) des expressions que vous pouvez utiliser pour spécifier des contraintes ou des conditions. DynamoDB Mapper fournit un Kotlin DSL idiomatique pour créer

des expressions. DSL Cela améliore la structure et la lisibilité de votre code et facilite également l'écriture d'expressions.

Cette section décrit la DSL syntaxe et fournit divers exemples.

Utiliser des expressions dans les opérations

Vous utilisez des expressions dans des opérations telles que `scan`, où elles filtrent les éléments renvoyés en fonction de critères que vous définissez. Pour utiliser des expressions avec DynamoDB Mapper, ajoutez le composant d'expression dans la demande d'opération.

L'extrait suivant montre un exemple d'expression de filtre utilisée dans une `scan` opération. Il utilise un argument `lambda` pour décrire les critères de filtre qui limitent les éléments à renvoyer à ceux dont la valeur de `year` attribut est 2001 :

```
val table = // A table instance.

table.scanPaginated {
    filter {
        attr("year") eq 2001
    }
}
```

L'exemple suivant montre une `query` opération qui prend en charge les expressions à deux endroits : le filtrage par clé de tri et le filtrage non par clé :

```
table.queryPaginated {
    keyCondition = KeyFilter(partitionKey = 1000) { sortKey startsWith "M" }
    filter {
        attr("year") eq 2001
    }
}
```

Le code précédent filtre les résultats selon ceux qui répondent aux trois critères :

- La valeur de l'attribut de clé de partition est 1000 - AND -
- La valeur de l'attribut clé de tri commence par la lettre M - AND -
- la valeur de l'attribut de l'année est 2001

Composants DSL

La DSL syntaxe expose plusieurs types de composants, décrits ci-dessous, que vous utilisez pour créer des expressions.

Attributs

La plupart des conditions font référence à des attributs, qui sont identifiés par leur clé ou leur chemin de document. Avec leDSK, vous pouvez créer toutes les références d'attributs à l'aide de la `attr` fonction et éventuellement apporter des modifications supplémentaires.

Le code suivant montre une série d'exemples de références d'attributs allant du plus simple au plus complexe, tels que le déréférencement de listes par index et le déréférencement de cartes par clé :

```
attr("foo")           // Refers to the value of top-level attribute `foo`.
attr("foo")[3]        // Refers to the value at index 3 in the list value of
                      // attribute `foo`.
attr("foo")[3]["bar"] // Refers to the value of key `bar` in the map value at
                      // index 3 of the list value of attribute `foo`.
```

Égalités et inégalités

Vous pouvez comparer les valeurs d'attribut d'une expression par égalités et inégalités. Vous pouvez comparer les valeurs d'attribut à des valeurs littérales ou à d'autres valeurs d'attribut. Les fonctions que vous utilisez pour définir les conditions sont les suivantes :

- `eq`: est égal à (équivalent à `==`)
- `neq`: n'est pas égal à (équivalent à `!=`)
- `gt`: est supérieur à (équivalent à `>`)
- `gte`: est supérieur ou égal à (équivalent à `>=`)
- `lt`: est inférieur à (équivalent à `<`)
- `lte`: est inférieur ou égal à (équivalent à `<=`)

Vous combinez la fonction de comparaison avec des arguments en utilisant la notation infixe, comme indiqué dans les exemples suivants :

```

attr("foo") eq 42           // Uses a literal. Specifies that the attribute value `foo`
    must be                 // equal to 42.

attr("bar") gte attr("baz") // Uses another attribute value. Specifies that the
    attribute               // value `bar` must be greater than or equal to the
                           // attribute value of `baz`.

```

Gammes et sets

Outre les valeurs uniques, vous pouvez comparer les valeurs des attributs à plusieurs valeurs dans des plages ou des ensembles. Vous utilisez la [isIn](#) fonction infix pour effectuer la comparaison, comme indiqué dans les exemples suivants :

```

attr("foo") isIn 0..99 // Specifies that the attribute value `foo` must be
    // in the range of `0` to `99` (inclusive).

attr("foo") isIn setOf( // Specifies that the attribute value `foo` must be
    "apple",             // one of `apple`, `banana`, or `cherry`.
    "banana",
    "cherry",
)

```

La `isIn` fonction fournit des surcharges pour les collections (par exemple `Set<String>`) et pour les limites que vous pouvez exprimer sous forme de Kotlin [ClosedRange<T>](#) (par exemple). [IntRange](#) Pour les limites que vous ne pouvez pas exprimer sous forme de `ClosedRange<T>` (comme des tableaux d'octets ou d'autres références d'attributs), vous pouvez utiliser la [isBetween](#) fonction :

```

val lowerBytes = byteArrayOf(0x48, 0x65, 0x6c) // Specifies that the attribute value
val upperBytes = byteArrayOf(0x6c, 0x6f, 0x21) // `foo` is between the values
attr("foo").isBetween(lowerBytes, upperBytes) // `0x48656c` and `0x6c6f21`

attr("foo").isBetween(attr("bar"), attr("baz")) // Specifies that the attribute value
    // `foo` is between the values of
    // attributes `bar` and `baz`.

```

Logique booléenne

Vous pouvez combiner des conditions individuelles ou les modifier à l'aide de la logique booléenne à l'aide des fonctions suivantes :

- **and**: chaque condition doit être vraie (équivalent à `&&`)
- **or**: au moins une condition doit être vraie (équivalent à `| |`)
- **not**: la condition donnée doit être fausse (équivalente à `!`)

Les exemples suivants illustrent chaque fonction :

```
and(                                // Both conditions must be met:
  attr("foo") eq "banana",         // * attribute value `foo` must equal `banana`
  attr("bar") isIn 0..99,          // * attribute value `bar` must be between
)                                    // 0 and 99 (inclusive)

or(                                  // At least one condition must be met:
  attr("foo") eq "cherry",         // * attribute value `foo` must equal `cherry`
  attr("bar") isIn 100..199,       // * attribute value `bar` must be between
)                                    // 100 and 199 (inclusive)

not(                                  // The attribute value `foo` must *not* be
  attr("baz") isIn setOf(          // one of `apple`, `banana`, or `cherry`.
    "apple",                       // Stated another way, the attribute value
    "banana",                       // must be *anything except* `apple`, `banana`,
    "cherry",                       // or `cherry`--including potentially a
  ),                                 // non-string value or no value at all.
)
```

Vous pouvez également combiner des conditions booléennes par des fonctions booléennes pour créer une logique imbriquée, comme indiqué dans l'expression suivante :

```
or(
  and(
    attr("foo") eq 123,
    attr("bar") eq "abc",
  ),
  and(
    attr("foo") eq 234,
    attr("bar") eq "bcd",
  ),
)
```

L'expression précédente filtre les résultats en fonction de ceux qui répondent à l'une des conditions suivantes :

- Ces deux conditions sont vraies :
 - foo la valeur de l'attribut est 123 - AND -
 - bar la valeur de l'attribut est « abc »
- Ces deux conditions sont vraies :
 - foo la valeur de l'attribut est 234 - AND -
 - bar la valeur de l'attribut est « bcd »

Cela équivaut à l'expression booléenne Kotlin suivante :

```
(foo == 123 && bar == "abc") || (foo == 234 && bar == "bcd")
```

Fonctions et propriétés

Les fonctions et propriétés suivantes fournissent des fonctionnalités d'expression supplémentaires :

- [contains](#): vérifie si la valeur d'un attribut de chaîne/liste contient une valeur donnée
- [exists](#): vérifie si un attribut est défini et contient une valeur quelconque (y compris null)
- [notExists](#): vérifie si un attribut n'est pas défini
- [isOfType](#): vérifie si la valeur d'un attribut est d'un type donné, tel qu'une chaîne, un nombre, un booléen, etc.
- [size](#): obtient la taille d'un attribut, comme le nombre d'éléments d'une collection ou la longueur d'une chaîne
- [startsWith](#): vérifie si la valeur d'un attribut de chaîne commence par une sous-chaîne donnée

Les exemples suivants montrent l'utilisation de fonctions et de propriétés supplémentaires que vous pouvez utiliser dans des expressions :

```
attr("foo") contains "apple" // Specifies that the attribute value `foo` must be
                             // a list that contains an `apple` element or a string
                             // which contains the substring `apple`.

attr("bar").exists()        // Specifies that the `bar` must exist and have a
                             // value (including potentially `null`).

attr("baz").size lt 100     // Specifies that the attribute value `baz` must have
                             // a size of less than 100.
```

```
attr("qux") isOfType AttributeType.String // Specifies that the attribute `qux`  
                                           // must have a string value.
```

Trier les filtres clés

Les expressions de filtre sur les clés de tri (comme dans le `keyCondition` paramètre de l'opération `query`) n'utilisent pas de valeurs d'attributs nommées. Pour utiliser une clé de tri dans un filtre, vous devez utiliser le mot-clé `sortKey` dans toutes les comparaisons. Le `sortKey` mot clé est remplacé `attr("<sort key name>")` comme indiqué dans les exemples suivants :

```
sortKey startsWith "abc" // The sort key attribute value must begin with the  
                        // substring `abc`.  
  
sortKey isIn 0..99      // The sort key attribute value must be between 0  
                        // and 99 (inclusive).
```

Vous ne pouvez pas combiner les filtres par clé de tri avec la logique booléenne et ils ne prennent en charge qu'un sous-ensemble des comparaisons décrites ci-dessus :

- [Égalités et inégalités](#) : toutes les comparaisons sont fondées
- [Plages et ensembles](#) : toutes les comparaisons sont prises en charge
- [Logique booléenne](#) : non prise en charge
- [Fonctions et propriétés](#) : seules les fonctions `startsWith` sont prises en charge

Exemples de code SDK pour Kotlin

Les exemples de code présentés dans cette rubrique vous montrent comment utiliser le AWS SDK pour Kotlin avec. AWS

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Certains services contiennent des exemples de catégories supplémentaires qui montrent comment tirer parti des bibliothèques ou des fonctions spécifiques au service.

Services

- [Exemples d'API Gateway utilisant le SDK pour Kotlin](#)
- [Exemples d'Aurora utilisant le SDK pour Kotlin](#)
- [Exemples d'Auto Scaling utilisant le SDK pour Kotlin](#)
- [Exemples d'Amazon Bedrock utilisant le SDK pour Kotlin](#)
- [CloudWatch exemples d'utilisation du SDK pour Kotlin](#)
- [CloudWatch Exemples de journaux utilisant le SDK pour Kotlin](#)
- [Exemples de fournisseurs d'identité Amazon Cognito utilisant le SDK pour Kotlin](#)
- [Exemples d'Amazon Comprehend utilisant le SDK pour Kotlin](#)
- [Exemples DynamoDB utilisant le SDK pour Kotlin](#)
- [EC2 Exemples Amazon utilisant le SDK pour Kotlin](#)
- [Exemples Amazon ECR utilisant le SDK pour Kotlin](#)
- [OpenSearch Exemples de services utilisant le SDK pour Kotlin](#)
- [EventBridge exemples d'utilisation du SDK pour Kotlin](#)
- [AWS Glue exemples d'utilisation du SDK pour Kotlin](#)
- [Exemples d'IAM utilisant le SDK pour Kotlin](#)

- [AWS IoT exemples d'utilisation du SDK pour Kotlin](#)
- [AWS IoT data exemples d'utilisation du SDK pour Kotlin](#)
- [Exemples d'Amazon Keyspaces utilisant le SDK pour Kotlin](#)
- [AWS KMS exemples d'utilisation du SDK pour Kotlin](#)
- [Exemples Lambda utilisant le SDK pour Kotlin](#)
- [MediaConvert exemples d'utilisation du SDK pour Kotlin](#)
- [Exemples d'Amazon Pinpoint utilisant le SDK pour Kotlin](#)
- [Exemples d'Amazon RDS utilisant le SDK pour Kotlin](#)
- [Exemples d'Amazon RDS Data Service utilisant le SDK pour Kotlin](#)
- [Exemples d'Amazon Redshift utilisant le SDK pour Kotlin](#)
- [Exemples d'Amazon Rekognition utilisant le SDK pour Kotlin](#)
- [Exemples d'enregistrement de domaine Route 53 à l'aide du SDK pour Kotlin](#)
- [Exemples d'Amazon S3 utilisant le SDK pour Kotlin](#)
- [SageMaker Exemples d'IA utilisant le SDK pour Kotlin](#)
- [Exemples de Secrets Manager utilisant le SDK pour Kotlin](#)
- [Exemples d'Amazon SES utilisant le SDK pour Kotlin](#)
- [Exemples d'Amazon SNS utilisant le SDK pour Kotlin](#)
- [Exemples Amazon SQS utilisant le SDK pour Kotlin](#)
- [Exemples de Step Functions utilisant le SDK pour Kotlin](#)
- [Support exemples d'utilisation du SDK pour Kotlin](#)
- [Exemples d'Amazon Translate utilisant le SDK pour Kotlin](#)

Exemples d'API Gateway utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec API Gateway.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Scénarios](#)

Scénarios

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

SDK pour Kotlin

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

Exemples d'Aurora utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Aurora.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Principes de base](#)
- [Actions](#)
- [Scénarios](#)

Principes de base

Principes de base

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 Kotlin

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](#).

```
/**
Before running this Kotlin 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-kotlin/latest/developer-guide/setup.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

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster 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 the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
*/

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = ""
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
            <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
```



```
        dbClusterGroupName - The database group name.
        dbParameterGroupFamily - The database parameter group name.
        dbInstanceClusterIdentifier - The database instance identifier.
        dbName - The database name.
        dbSnapshotIdentifier - The snapshot identifier.
        secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()

    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
```

```
describeDbClusterParameters(dbClusterGroupName, -1)

println("7. Get a list of allowed engine versions")
getAllowedClusterEngines(dbParameterGroupFamily)

println("8. Create an Aurora DB cluster database")
val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
println("The ARN of the cluster is $arnClusterVal")

println("9. Wait for DB instance to be ready")
waitForClusterInstanceReady(dbInstanceClusterIdentifier)

println("10. Get a list of instance classes available for the selected engine")
val instanceClass = getListInstanceClasses()

println("11. Create a database instance in the cluster.")
val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
dbInstanceClusterIdentifier, instanceClass)
println("The ARN of the database is $clusterDBARN")

println("12. Wait for DB instance to be ready")
waitDBAuroraInstanceReady(dbInstanceIdentifier)

println("13. Create a snapshot")
createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

println("14. Wait for DB snapshot to be ready")
waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)

println("15. Delete the DB instance")
deleteDBInstance(dbInstanceIdentifier)

println("16. Delete the DB cluster")
deleteCluster(dbInstanceClusterIdentifier)

println("17. Delete the DB cluster group")
if (clusterDBARN != null) {
    deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
}
println("The Scenario has successfully completed.")
}

@Throws(InterruptedExcetion::class)
```

```

suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                }
                if (index == listSize && !didFind) {
                    // Went through the entire list and did not find the
database ARN.
                    isDataDel = true
                }
                delay(slTime * 1000)
                index++
            }
        }
    }
    val clusterParameterGroupRequest =
        DeleteDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupName
        }

    rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
    println("$dbClusterGroupName was deleted.")
}
}

```

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
        ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
```

```

        val response = rdsClient.describeDbClusterSnapshots(snapshotRequest)
        val snapshotList = response.dbClusterSnapshots
        if (snapshotList != null) {
            for (snapshot in snapshotList) {
                snapshotReadyStr = snapshot.status.toString()
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true
                } else {
                    println(".")
                    delay(slTime * 5000)
                }
            }
        }
    }
}
println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
    ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
}

```

```
var endpoint = ""
RdsClient { region = "us-west-2" }.use { rdsClient ->
    while (!instanceReady) {
        val response = rdsClient.describeDbInstances(instanceRequest)
        response.dbInstances?.forEach { instance ->
            instanceReadyStr = instance.dbInstanceStatus.toString()
            if (instanceReadyStr.contains("available")) {
                endpoint = instance.endpoint?.address.toString()
                instanceReady = true
            } else {
                print(".")
                delay(sleepTime * 1000)
            }
        }
    }
}
println("Database instance is available! The connection endpoint is $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
}
```

```
    }
    var instanceClass = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
```

```

    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }
}

```



```

    }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
    successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbClusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbClusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
    rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
    auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
    paraName.compareTo("auto_increment_increment ") == 0) {

```

```

        println("*** The parameter name is $paraName")
        println("*** The parameter value is ${para.parameterValue}")
        println("*** The parameter data type is ${para.dataType}")
        println("*** The parameter description is ${para.description}")
        println("*** The parameter allowed values is
    ${para.allowedValues}")
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}

suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
    ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

```

```
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engine0b ->
            println("The name of the DB parameter group family for the database
engine is ${engine0b.dbParameterGroupFamily}")
            println("The name of the database engine ${engine0b.engine}")
            println("The version number of the database engine
${engine0b.engineVersion}")
        }
    }
}
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CréerDBCluster](#)
 - [CréerDBClusterParameterGroup](#)
 - [Créer un DBCluster instantané](#)
 - [CréerDBInstance](#)
 - [SuppressionDBCluster](#)
 - [SuppressionDBClusterParameterGroup](#)
 - [SuppressionDBInstance](#)
 - [Décrivez DBCluster ParameterGroups](#)
 - [Décrire DBCluster les paramètres](#)
 - [Décrire les DBCluster instantanés](#)
 - [Décrivez DBClusters](#)
 - [Décrire DBEngine les versions](#)

- [Décrivez DBInstances](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBClusterParameterGroup](#)

Actions

CreateDBCluster

L'exemple de code suivant montre comment utiliser `CreateDBCluster`.

SDK pour Kotlin

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](#).

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Pour plus de détails sur l'API, voir [Create DBCluster](#) in AWS SDK for Kotlin API reference.

CreateDBClusterParameterGroup

L'exemple de code suivant montre comment utiliser `CreateDBClusterParameterGroup`.

SDK pour Kotlin

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](#).

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
    ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Pour plus de détails sur l'API, voir [Create DBCluster ParameterGroup](#) in AWS SDK for Kotlin API reference.

CreateDBClusterSnapshot

L'exemple de code suivant montre comment utiliser `CreateDBClusterSnapshot`.

SDK pour Kotlin

 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](#).

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }


    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
        ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Pour plus de détails sur l'API, voir [Créer un DBCluster instantané](#) dans le AWS SDK pour la référence de l'API Kotlin.

CreateDBInstance

L'exemple de code suivant montre comment utiliser CreateDBInstance.

SDK pour Kotlin

 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](#).

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Pour plus de détails sur l'API, voir [Create DBInstance](#) in AWS SDK for Kotlin API reference.

DeleteDBCluster

L'exemple de code suivant montre comment utiliser `DeleteDBCluster`.

SDK pour Kotlin

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](#).

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }
}
```

```

RdsClient { region = "us-west-2" }.use { rdsClient ->
    rdsClient.deleteDbCluster(deleteDbClusterRequest)
    println("$dbInstanceClusterIdentifier was deleted!")
}
}

```

- Pour plus de détails sur l'API, voir [Supprimer DBCluster dans le AWS SDK](#) pour la référence de l'API Kotlin.

DeleteDBClusterParameterGroup

L'exemple de code suivant montre comment utiliser `DeleteDBClusterParameterGroup`.

SDK pour Kotlin

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](#).

```

@Throws(InterruptedExcetion::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1

```



```

        if (instanceList != null) {
            for (instance in instanceList) {
                instanceARN = instance.dbInstanceArn.toString()
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    println("$clusterDBARN still exists")
                    didFind = true
                }
            }
            if (index == listSize && !didFind) {
                // Went through the entire list and did not find the
database ARN.
                isDataDel = true
            }
            delay(slTime * 1000)
            index++
        }
    }
}
val clusterParameterGroupRequest =
    DeleteDbClusterParameterGroupRequest {
        dbClusterParameterGroupName = dbClusterGroupName
    }

rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
println("$dbClusterGroupName was deleted.")
}
}

```

- Pour plus de détails sur l'API, voir [Supprimer DBCluster ParameterGroup dans le AWS SDK](#) pour la référence de l'API Kotlin.

DeleteDBInstance

L'exemple de code suivant montre comment utiliser DeleteDBInstance.

SDK pour Kotlin

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](#).

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
        ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Pour plus de détails sur l'API, voir [Supprimer DBInstance dans le AWS SDK](#) pour la référence de l'API Kotlin.

DescribeDBClusterParameterGroups

L'exemple de code suivant montre comment utiliser `DescribeDBClusterParameterGroups`.

SDK pour Kotlin

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](#).

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
```

```
        println("The group name is ${group.dbClusterParameterGroupName}")
        println("The group ARN is ${group.dbClusterParameterGroupArn}")
    }
}
}
```

- Pour plus de détails sur l'API, voir [Description DBCluster ParameterGroups](#) dans le AWS SDK pour la référence de l'API Kotlin.

DescribeDBClusterParameters

L'exemple de code suivant montre comment utiliser `DescribeDBClusterParameters`.

SDK pour Kotlin

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](#).

```
suspend fun describeDbClusterParameters(
    dbClusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbClusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
```

```

        val response =
            rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
            response.parameters?.forEach { para ->
                // Only print out information about either auto_increment_offset or
                auto_increment_increment.
                val paraName = para.parameterName
                if (paraName != null) {
                    if (paraName.compareTo("auto_increment_offset") == 0 ||
                        paraName.compareTo("auto_increment_increment ") == 0) {
                        println("*** The parameter name is $paraName")
                        println("*** The parameter value is ${para.parameterValue}")
                        println("*** The parameter data type is ${para.dataType}")
                        println("*** The parameter description is ${para.description}")
                        println("*** The parameter allowed values is
                        ${para.allowedValues}")
                    }
                }
            }
        }
    }
}

```

- Pour plus de détails sur l'API, voir [Description DBCluster des paramètres](#) dans le AWS SDK pour la référence de l'API Kotlin.

DescribeDBClusterSnapshots

L'exemple de code suivant montre comment utiliser DescribeDBClusterSnapshots.

SDK pour Kotlin

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](#).

```

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {

```

```

var snapshotReady = false
var snapshotReadyStr: String
println("Waiting for the snapshot to become available.")

val snapshotsRequest =
    DescribeDbClusterSnapshotsRequest {
        dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        dbClusterIdentifier = dbInstanceClusterIdentifier
    }

RdsClient { region = "us-west-2" }.use { rdsClient ->
    while (!snapshotReady) {
        val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
        val snapshotList = response.dbClusterSnapshots
        if (snapshotList != null) {
            for (snapshot in snapshotList) {
                snapshotReadyStr = snapshot.status.toString()
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true
                } else {
                    println(".")
                    delay(slTime * 5000)
                }
            }
        }
    }
}
println("The Snapshot is available!")
}

```

- Pour plus de détails sur l'API, consultez la section [Description DBCluster des instantanés](#) dans le AWS SDK pour la référence de l'API Kotlin.

DescribeDBClusters

L'exemple de code suivant montre comment utiliser `DescribeDBClusters`.

SDK pour Kotlin

 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](#).

```
suspend fun describeDbClusterParameters(
    dbClusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbClusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
            rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
            auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
                    paraName.compareTo("auto_increment_increment ") == 0) {
                    println("**** The parameter name is $paraName")
                    println("**** The parameter value is ${para.parameterValue}")
                    println("**** The parameter data type is ${para.dataType}")
                    println("**** The parameter description is ${para.description}")
                    println("**** The parameter allowed values is
                    ${para.allowedValues}")
                }
            }
        }
    }
}
```

```
    }  
  }  
}
```

- Pour plus de détails sur l'API, voir [Description DBClusters](#) dans le AWS SDK pour la référence de l'API Kotlin.

DescribeDBEngineVersions

L'exemple de code suivant montre comment utiliser `DescribeDBEngineVersions`.

SDK pour Kotlin

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](#).

```
// Get a list of allowed engine versions.  
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {  
    val versionsRequest =  
        DescribeDbEngineVersionsRequest {  
            dbParameterGroupFamily = dbParameterGroupFamilyVal  
            engine = "aurora-mysql"  
        }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.describeDbEngineVersions(versionsRequest)  
        response.dbEngineVersions?.forEach { dbEngine ->  
            println("The engine version is ${dbEngine.engineVersion}")  
            println("The engine description is ${dbEngine.dbEngineDescription}")  
        }  
    }  
}
```

- Pour plus de détails sur l'API, voir [Description des DBEngine versions](#) dans le AWS SDK pour la référence de l'API Kotlin.

DescribeDBInstances

L'exemple de code suivant montre comment utiliser `DescribeDBInstances`.

SDK pour Kotlin

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](#).

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}
```


- Pour plus de détails sur l'API, voir [Description DBInstances](#) dans le AWS SDK pour la référence de l'API Kotlin.

ModifyDBClusterParameterGroup

L'exemple de code suivant montre comment utiliser `ModifyDBClusterParameterGroup`.

SDK pour Kotlin

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](#).

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
successfully modified")
    }
}
```

- Pour plus de détails sur l'API, voir [Modifier DBCluster ParameterGroup](#) dans le AWS SDK pour la référence de l'API Kotlin.

Scénarios

Créer un outil de suivi des éléments de travail sans serveur Aurora

L'exemple de code suivant montre comment créer une application Web qui suit les éléments de travail dans une base de données Amazon Aurora Serverless et utilise Amazon Simple Email Service (Amazon SES) pour envoyer des rapports.

SDK pour Kotlin

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](#).

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Exemples d'Auto Scaling utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Auto Scaling.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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 Kotlin

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](#).

```
suspend fun main(args: Array<String>) {
    val usage = ""
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
```

```
    launchTemplateName - The name of the launch template.
    serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-linked
role that the Auto Scaling group uses.
    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances in
the Auto Scaling group can be created.
    ""

if (args.size != 4) {
    println(usage)
    exitProcess(1)
}

val groupName = args[0]
val launchTemplateName = args[1]
val serviceLinkedRoleARN = args[2]
val vpcZoneId = args[3]

println("**** Create an Auto Scaling group named $groupName")
createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
vpcZoneId)

println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
delay(60000)

val instanceId = getSpecificAutoScaling(groupName)
if (instanceId.compareTo("") == 0) {
    println("Error - no instance Id value")
    exitProcess(1)
} else {
    println("The instance Id value is $instanceId")
}

println("**** Describe Auto Scaling with the Id value $instanceId")
describeAutoScalingInstance(instanceId)

println("**** Enable metrics collection $instanceId")
enableMetricsCollection(groupName)

println("**** Update an Auto Scaling group to maximum size of 3")
updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

println("**** Describe all Auto Scaling groups to show the current state of the
groups")
```

```

describeAutoScalingGroups(groupName)

println("**** Describe account details")
describeAccountLimits()

println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
delay(60000)

println("**** Set desired capacity to 2")
setDesiredCapacity(groupName)

println("**** Get the two instance Id values and state")
getAutoScalingGroups(groupName)

println("**** List the scaling activities that have occurred for the group")
describeScalingActivities(groupName)

println("**** Terminate an instance in the Auto Scaling group")
terminateInstanceInAutoScalingGroup(instanceId)

println("**** Stop the metrics collection")
disableMetricsCollection(groupName)

println("**** Delete the Auto Scaling group")
deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
${group.healthCheckType}")
        }
    }
}
}

```

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
            response.activities?.forEach { activity ->
                println("The activity Id is ${activity.activityId}")
                println("The activity details are ${activity.details}")
            }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
            response.autoScalingGroups?.forEach { group ->
                println("The group name is ${group.autoScalingGroupName}")
                println("The group ARN is ${group.autoScalingGroupArn}")
                group.instances?.forEach { instance ->
                    println("The instance id is ${instance.instanceId}")
                }
            }
    }
}
```

```
        println("The lifecycle state is " + instance.lifecycleState)
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
    }
}
```

```
        autoScalingClient.waitForGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitForGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }
}
```



```
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
        autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
        autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
}
```

```
        return instanceId
    }

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
        autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
        ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
        ${response.numberOfWorkingAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Actions

CreateAutoScalingGroup

L'exemple de code suivant montre comment utiliser `CreateAutoScalingGroup`.

SDK pour Kotlin

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](#).

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
```

```

    CreateAutoScalingGroupRequest {
        autoScalingGroupName = groupName
        availabilityZones = listOf("us-east-1a")
        launchTemplate = templateSpecification
        maxSize = 1
        minSize = 1
        vpcZoneIdentifier = vpcZoneIdVal
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
    }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateAutoScalingGroup](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteAutoScalingGroup

L'exemple de code suivant montre comment utiliser `DeleteAutoScalingGroup`.

SDK pour Kotlin

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](#).

```

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =

```

```
DeleteAutoScalingGroupRequest {
    autoScalingGroupName = groupName
    forceDelete = true
}

AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
    autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
    println("You successfully deleted $groupName")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAutoScalingGroup](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeAutoScalingGroups

L'exemple de code suivant montre comment utiliser `DescribeAutoScalingGroups`.

SDK pour Kotlin

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](#).

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
            }
        }
    }
}
```

```
        println("The lifecycle state is " + instance.lifecycleState)
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAutoScalingGroups](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeAutoScalingInstances

L'exemple de code suivant montre comment utiliser `DescribeAutoScalingInstances`.

SDK pour Kotlin

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](#).

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
            response.autoScalingInstances?.forEach { group ->
                println("The instance lifecycle state is: ${group.lifecycleState}")
            }
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAutoScalingInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeScalingActivities

L'exemple de code suivant montre comment utiliser `DescribeScalingActivities`.

SDK pour Kotlin

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](#).

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
                ${group.healthCheckType}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeScalingActivities](#) à la section AWS SDK pour la référence de l'API Kotlin.

DisableMetricsCollection

L'exemple de code suivant montre comment utiliser `DisableMetricsCollection`.

SDK pour Kotlin

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](#).

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DisableMetricsCollection](#) à la section AWS SDK pour la référence de l'API Kotlin.

EnableMetricsCollection

L'exemple de code suivant montre comment utiliser `EnableMetricsCollection`.

SDK pour Kotlin

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](#).

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
```



```
        autoScalingGroupName = groupName
        metrics = listOf("GroupMaxSize")
        granularity = "1Minute"
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [EnableMetricsCollection](#) à la section AWS SDK pour la référence de l'API Kotlin.

SetDesiredCapacity

L'exemple de code suivant montre comment utiliser `SetDesiredCapacity`.

SDK pour Kotlin

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](#).

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SetDesiredCapacity](#) à la section AWS SDK pour la référence de l'API Kotlin.

TerminateInstanceInAutoScalingGroup

L'exemple de code suivant montre comment utiliser `TerminateInstanceInAutoScalingGroup`.

SDK pour Kotlin

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](#).

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstanceInAutoScalingGroup](#) à la section AWS SDK pour la référence de l'API Kotlin.

UpdateAutoScalingGroup

L'exemple de code suivant montre comment utiliser `UpdateAutoScalingGroup`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group $groupName")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateAutoScalingGroup](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples d'Amazon Bedrock utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec 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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

ListFoundationModels

L'exemple de code suivant montre comment utiliser `ListFoundationModels`.

SDK pour Kotlin

Note

Il y en a plus 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 fondations Amazon Bedrock disponibles.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient { region = "us-east-1" }.use { bedrockClient ->
        val response =
            bedrockClient.listFoundationModels(ListFoundationModelsRequest {})
        response.modelSummaries?.forEach { model ->
            println("=====")
            println(" Model ID: ${model.modelId}")
            println("-----")
        }
    }
}
```

```
        println(" Name: ${model.modelName}")
        println(" Provider: ${model.providerName}")
        println(" Input modalities: ${model.inputModalities}")
        println(" Output modalities: ${model.outputModalities}")
        println(" Supported customizations: ${model.customizationsSupported}")
        println(" Supported inference types: ${model.inferenceTypesSupported}")
        println("-----\n")
    }
    return response.modelSummaries
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListFoundationModels](#) à la section AWS SDK pour la référence de l'API Kotlin.

CloudWatch exemples d'utilisation du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec CloudWatch

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 démarrer avec CloudWatch.

SDK pour Kotlin

 Note

Il y en a plus 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 Kotlin 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-kotlin/latest/developer-guide/setup.html
*/
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <namespace>
        Where:
            namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listMetricsPaginated(request)
    }
}
```

```
        .transform { it.metrics?.forEach { obj -> emit(obj) } }
        .collect { obj ->
            println("Name is ${obj.metricName}")
            println("Namespace is ${obj.namespace}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListMetrics](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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 Kotlin

Note

Il y en a plus 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 CloudWatch les fonctionnalités.

```
/**
Before running this Kotlin 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-kotlin/latest/developer-guide/setup.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 Kotlin code example performs the following tasks:

1. List available namespaces from Amazon CloudWatch. Select a namespace from the
list.
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.
*/

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = ""
    Usage:
```



```
<myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
<settings> <metricImage>
```

Where:

myDate - The start date to use to get metric statistics. (For example, 2023-01-11T18:35:24.00Z.)

costDateWeek - The start date to use to get AWS Billing and Cost Management statistics. (For example, 2023-01-11T18:35:24.00Z.)

dashboardName - The name of the dashboard to create.

dashboardJson - The location of a JSON file to use to create a dashboard. (See Readme file.)

dashboardAdd - The location of a JSON file to use to update a dashboard. (See Readme file.)

settings - The location of a JSON file from which various values are read. (See Readme file.)

metricImage - The location of a BMP file that is used to create a graph.

```
"""
```

```
if (args.size != 7) {
    println(usage)
    System.exit(1)
}
```

```
val myDate = args[0]
val costDateWeek = args[1]
val dashboardName = args[2]
val dashboardJson = args[3]
val dashboardAdd = args[4]
val settings = args[5]
var metricImage = args[6]
val dataPoint = "10.0".toDouble()
val in0b = Scanner(System.`in`)
```

```
println(DASHES)
println("Welcome to the Amazon CloudWatch example scenario.")
println(DASHES)
```

```
println(DASHES)
println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
val list: ArrayList<String> = listNameSpaces()
for (z in 0..4) {
    println("    ${z + 1}. ${list[z]}")
```

```
}

var selectedNamespace: String
var selectedMetrics = ""
var num = inOb.nextLine().toInt()
println("You selected $num")

if (1 <= num && num <= 5) {
    selectedNamespace = list[num - 1]
} else {
    println("You did not select a valid option.")
    exitProcess(1)
}

println("You selected $selectedNamespace")
println(DASHES)

println(DASHES)
println("2. List available metrics within the selected namespace and select one
from the list.")
val metList = listMets(selectedNamespace)
for (z in 0..4) {
    println("    ${ z + 1}. ${metList?.get(z)}")
}
num = inOb.nextLine().toInt()
if (1 <= num && num <= 5) {
    selectedMetrics = metList!![num - 1]
} else {
    println("You did not select a valid option.")
    System.exit(1)
}
println("You selected $selectedMetrics")
val myDimension = getSpecificMet(selectedNamespace)
if (myDimension == null) {
    println("Error - Dimension is null")
    exitProcess(1)
}
println(DASHES)

println(DASHES)
println("3. Get statistics for the selected metric over the last day.")
val metricOption: String
val statTypes = ArrayList<String>()
statTypes.add("SampleCount")
statTypes.add("Average")
```

```
statTypes.add("Sum")
statTypes.add("Minimum")
statTypes.add("Maximum")

for (t in 0..4) {
    println("    ${t + 1}. ${statTypes[t]}")
}
println("Select a metric statistic by entering a number from the preceding
list:")
num = in0b.nextLine().toInt()
if (1 <= num && num <= 5) {
    metricOption = statTypes[num - 1]
} else {
    println("You did not select a valid option.")
    exitProcess(1)
}
println("You selected $metricOption")
getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics, metricOption,
myDate, myDimension)
println(DASHES)

println(DASHES)
println("4. Get CloudWatch estimated billing for the last week.")
getMetricStatistics(costDateWeek)
println(DASHES)

println(DASHES)
println("5. Create a new CloudWatch dashboard with metrics.")
createDashboardWithMetrics(dashboardName, dashboardJson)
println(DASHES)

println(DASHES)
println("6. List dashboards using a paginator.")
listDashboards()
println(DASHES)

println(DASHES)
println("7. Create a new custom metric by adding data to it.")
createNewCustomMetric(dataPoint)
println(DASHES)

println(DASHES)
println("8. Add an additional metric to the dashboard.")
addMetricToDashboard(dashboardAdd, dashboardName)
```

```
println(DASHES)

println(DASHES)
println("9. Create an alarm for the custom metric.")
val alarmName: String = createAlarm(settings)
println(DASHES)

println(DASHES)
println("10. Describe 10 current alarms.")
describeAlarms()
println(DASHES)

println(DASHES)
println("11. Get current data for the new custom metric.")
getCustomMetricData(settings)
println(DASHES)

println(DASHES)
println("12. Push data into the custom metric to trigger the alarm.")
addMetricDataForAlarm(settings)
println(DASHES)

println(DASHES)
println("13. Check the alarm state using the action DescribeAlarmsForMetric.")
checkForMetricAlarm(settings)
println(DASHES)

println(DASHES)
println("14. Get alarm history for the new alarm.")
getAlarmHistory(settings, myDate)
println(DASHES)

println(DASHES)
println("15. Add an anomaly detector for the custom metric.")
addAnomalyDetector(settings)
println(DASHES)

println(DASHES)
println("16. Describe current anomaly detectors.")
describeAnomalyDetectors(settings)
println(DASHES)

println(DASHES)
println("17. Get a metric image for the custom metric.")
```

```

    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }
}

```

```
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
                "AWS/Billing",
                "EstimatedCharges",
                "Currency",
                "USD"
            ]
        ]
    }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
```

```
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }
}
```

```
    }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}

suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient {
        credentialsProvider = EnvironmentCredentialsProvider()
        region = "us-east-1"
    }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
```



```

        println("No alarm history data found for $alarmNameVal.")
    } else {
        for (item in historyItems) {
            println("History summary ${item.historySummary}")
            println("Time stamp: ${item.timestamp}")
        }
    }
}
}
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {

```

```
// Read values from the JSON file.
val parser = JsonFactory().createParser(File(fileName))
val rootNode = ObjectMapper().readTree<JsonNode>(parser)
val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
val customMetricName = rootNode.findValue("customMetricName").asText()

// Set an Instant object.
val time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
val instant = Instant.parse(time)
val datum =
    MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1001.00
        timestamp =
            aws.smithy.kotlin.runtime.time
                .Instant(instant)
    }

val datum2 =
    MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1002.00
        timestamp =
            aws.smithy.kotlin.runtime.time
                .Instant(instant)
    }

val metricDataList = ArrayList<MetricDatum>()
metricDataList.add(datum)
metricDataList.add(datum2)

val request =
    PutMetricDataRequest {
        namespace = customMetricNamespace
        metricData = metricDataList
    }

CloudWatchClient { region = "us-east-1" }.use { cwClient ->
    cwClient.putMetricData(request)
    println("Added metric values for for metric $customMetricName")
}
```

```
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQuery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQuery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
        }
}
```

```

        scanBy = ScanBy.TimestampDescending
        startTime =
            aws.smithy.kotlin.runtime.time
                .Instant(nowDate)
        endTime =
            aws.smithy.kotlin.runtime.time
                .Instant(date2)
        metricDataQueries = dq
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}

suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
}

```

```
val emailTopic = rootNode.findValue("emailTopic").asText()
val accountId = rootNode.findValue("accountId").asText()
val region2 = rootNode.findValue("region").asText()

// Create a List for alarm actions.
val alarmActionObs: MutableList<String> = ArrayList()
alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
val alarmRequest =
    PutMetricAlarmRequest {
        alarmActions = alarmActionObs
        alarmDescription = "Example metric alarm"
        alarmName = alarmNameVal
        comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
        threshold = 100.00
        metricName = customMetricName
        namespace = customMetricNamespace
        evaluationPeriods = 1
        period = 10
        statistic = Statistic.Maximum
        datapointsToAlarm = 1
        treatMissingData = "ignore"
    }

CloudWatchClient { region = "us-east-1" }.use { cwClient ->
    cwClient.putMetricAlarm(alarmRequest)
    println("$alarmNameVal was successfully created!")
    return alarmNameVal
}

suspend fun addMetricToDashboard(
    fileNameVal: String,
    dashboardNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}
```

```
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension =
        Dimension {
            name = "UNIQUE_PAGES"
            value = "URLS"
        }

    // Set an Instant object.
    val time =
        ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = "PAGES_VISITED"
            unit = StandardUnit.None
            value = dataPoint
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
            dimensions = listOf(dimension)
        }

    val request =
        PutMetricDataRequest {
            namespace = "SITE/TRAFFIC"
            metricData = listOf(datum)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

```
    }
  }
}

suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}

fun readFileAsString(file: String): String =
    String(Files.readAllBytes(Paths.get(file)))

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension =
        Dimension {
            name = "Currency"
            value = "USD"
        }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)
}
```

```

val statisticsRequest =
    GetMetricStatisticsRequest {
        metricName = "EstimatedCharges"
        namespace = "AWS/Billing"
        dimensions = dimensionList
        statistics = listOf(Statistic.Maximum)
        startTime =
            aws.smithy.kotlin.runtime.time
                .Instant(start)
        endTime =
            aws.smithy.kotlin.runtime.time
                .Instant(endDate)
        period = 86400
    }
CloudWatchClient { region = "us-east-1" }.use { cwClient ->
    val response = cwClient.getMetricStatistics(statisticsRequest)
    val data: List<Datapoint>? = response.datapoints
    if (data != null) {
        if (!data.isEmpty()) {
            for (datapoint in data) {
                println("Timestamp: ${datapoint.timestamp} Maximum value:
${datapoint.maximum}")
            }
        } else {
            println("The returned data list is empty")
        }
    }
}
}

suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time

```



```

        .Instant(endDate)
    startTime =
        aws.smithy.kotlin.runtime.time
            .Instant(start)
    dimensions = listOf(myDimension)
    metricName = metVal
    namespace = nameSpaceVal
    period = 86400
    statistics = listOf(Statistic.fromValue(metricOption))
}

CloudWatchClient { region = "us-east-1" }.use { cwClient ->
    val response = cwClient.getMetricStatistics(statisticsRequest)
    val data = response.datapoints
    if (data != null) {
        if (data.isNotEmpty()) {
            for (datapoint in data) {
                println("Timestamp: ${datapoint.timestamp} Maximum value:
${datapoint.maximum}")
            }
        } else {
            println("The returned data list is empty")
        }
    }
}
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
}

```

```
suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la AWS Référence de l'API de SDK pour Kotlin.
 - [DeleteAlarms](#)
 - [DeleteAnomalyDetector](#)
 - [DeleteDashboards](#)
 - [DescribeAlarmHistory](#)
 - [DescribeAlarms](#)
 - [DescribeAlarmsForMetric](#)
 - [DescribeAnomalyDetectors](#)

- [GetMetricData](#)
- [GetMetricStatistics](#)
- [GetMetricWidgetImage](#)
- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

Actions

DeleteAlarms

L'exemple de code suivant montre comment utiliser `DeleteAlarms`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAlarms](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteAnomalyDetector

L'exemple de code suivant montre comment utiliser `DeleteAnomalyDetector`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAnomalyDetector](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteDashboards

L'exemple de code suivant montre comment utiliser `DeleteDashboards`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDashboards](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeAlarmHistory

L'exemple de code suivant montre comment utiliser `DescribeAlarmHistory`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getAlarmHistory(
```

```
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient {
        credentialsProvider = EnvironmentCredentialsProvider()
        region = "us-east-1"
    }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAlarmHistory](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeAlarms

L'exemple de code suivant montre comment utiliser `DescribeAlarms`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAlarms](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeAlarmsForMetric

L'exemple de code suivant montre comment utiliser `DescribeAlarmsForMetric`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}
```


- Pour plus de détails sur l'API, reportez-vous [DescribeAlarmsForMetric](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeAnomalyDetectors

L'exemple de code suivant montre comment utiliser `DescribeAnomalyDetectors`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAnomalyDetectors](#) à la section AWS SDK pour la référence de l'API Kotlin.

DisableAlarmActions

L'exemple de code suivant montre comment utiliser `DisableAlarmActions`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DisableAlarmActions](#) à la section AWS SDK pour la référence de l'API Kotlin.

EnableAlarmActions

L'exemple de code suivant montre comment utiliser `EnableAlarmActions`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun enableActions(alarm: String) {
```

```
val request =
    EnableAlarmActionsRequest {
        alarmNames = listOf(alarm)
    }

CloudWatchClient { region = "us-east-1" }.use { cwClient ->
    cwClient.enableAlarmActions(request)
    println("Successfully enabled actions on alarm $alarm")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [EnableAlarmActions](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetMetricData

L'exemple de code suivant montre comment utiliser `GetMetricData`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
```

```
        ChronoUnit.MINUTES,
    )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQuery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQuery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
```

```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetMetricData](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetMetricStatistics

L'exemple de code suivant montre comment utiliser `GetMetricStatistics`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getAndDisplayMetricStatistics(  
    namespaceVal: String,  
    metVal: String,  
    metricOption: String,  
    date: String,  
    myDimension: Dimension,  
) {  
    val start = Instant.parse(date)  
    val endDate = Instant.now()  
    val statisticsRequest =  
        GetMetricStatisticsRequest {  
            endTime =  
                aws.smithy.kotlin.runtime.time  
                    .Instant(endDate)  
            startTime =  
                aws.smithy.kotlin.runtime.time  
                    .Instant(start)  
            dimensions = listOf(myDimension)  
            metricName = metVal  
            namespace = namespaceVal  
            period = 86400  
            statistics = listOf(Statistic.fromValue(metricOption))  
        }  
}
```

```

    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [GetMetricStatistics](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetMetricWidgetImage

L'exemple de code suivant montre comment utiliser `GetMetricWidgetImage`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,

```

```

        "width": 1400,
        "height": 600,
        "metrics": [
            [
                "AWS/Billing",
                "EstimatedCharges",
                "Currency",
                "USD"
            ]
        ]
    }""")

val imageRequest =
    GetMetricWidgetImageRequest {
        metricWidget = myJSON
    }

CloudWatchClient { region = "us-east-1" }.use { cwClient ->
    val response = cwClient.getMetricWidgetImage(imageRequest)
    val bytes = response.metricWidgetImage
    if (bytes != null) {
        File(fileName).writeBytes(bytes)
    }
}
println("You have successfully written data to $fileName")
}

```

- Pour plus de détails sur l'API, reportez-vous [GetMetricWidgetImage](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListDashboards

L'exemple de code suivant montre comment utiliser `ListDashboards`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDashboards](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListMetrics

L'exemple de code suivant montre comment utiliser `ListMetrics`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
}
```



```
    }  
  }  
  return metList  
}
```

- Pour plus de détails sur l'API, reportez-vous [ListMetrics](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutAnomalyDetector

L'exemple de code suivant montre comment utiliser `PutAnomalyDetector`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun addAnomalyDetector(fileName: String?) {  
    // Read values from the JSON file.  
    val parser = JsonFactory().createParser(File(fileName))  
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)  
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()  
    val customMetricName = rootNode.findValue("customMetricName").asText()  
  
    val singleMetricAnomalyDetectorVal =  
        SingleMetricAnomalyDetector {  
            metricName = customMetricName  
            namespace = customMetricNamespace  
            stat = "Maximum"  
        }  
  
    val anomalyDetectorRequest =  
        PutAnomalyDetectorRequest {  
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal  
        }  
  
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
```

```
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutAnomalyDetector](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutDashboard

L'exemple de code suivant montre comment utiliser PutDashboard.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

```
    }  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [PutDashboard](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutMetricAlarm

L'exemple de code suivant montre comment utiliser `PutMetricAlarm`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun putMetricAlarm(  
    alarmNameVal: String,  
    instanceIdVal: String,  
) {  
    val dimension0b =  
        Dimension {  
            name = "InstanceId"  
            value = instanceIdVal  
        }  
  
    val request =  
        PutMetricAlarmRequest {  
            alarmName = alarmNameVal  
            comparisonOperator = ComparisonOperator.GreaterThanThreshold  
            evaluationPeriods = 1  
            metricName = "CPUUtilization"  
            namespace = "AWS/EC2"  
            period = 60  
            statistic = Statistic.fromValue("Average")  
        }  
}
```

```

        threshold = 70.0
        actionsEnabled = false
        alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
utilization exceeds 70%"
        unit = StandardUnit.fromValue("Seconds")
        dimensions = listOf(dimension0b)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [PutMetricAlarm](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutMetricData

L'exemple de code suivant montre comment utiliser PutMetricData.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
        ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
}

```

```
val datum =
    MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1001.00
        timestamp =
            aws.smithy.kotlin.runtime.time
                .Instant(instant)
    }

val datum2 =
    MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1002.00
        timestamp =
            aws.smithy.kotlin.runtime.time
                .Instant(instant)
    }

val metricDataList = ArrayList<MetricDatum>()
metricDataList.add(datum)
metricDataList.add(datum2)

val request =
    PutMetricDataRequest {
        namespace = customMetricNamespace
        metricData = metricDataList
    }

CloudWatchClient { region = "us-east-1" }.use { cwClient ->
    cwClient.putMetricData(request)
    println("Added metric values for for metric $customMetricName")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutMetricData](#) à la section AWS SDK pour la référence de l'API Kotlin.

CloudWatch Exemples de journaux utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin 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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { logs ->
```

```
logs.deleteSubscriptionFilter(request)
println("Successfully deleted CloudWatch logs subscription filter named
$filter")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSubscriptionFilter](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeSubscriptionFilters

L'exemple de code suivant montre comment utiliser `DescribeSubscriptionFilters`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name ${filter.filterName} pattern
            ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSubscriptionFilters](#) à la section AWS SDK pour la référence de l'API Kotlin.

StartLiveTail

L'exemple de code suivant montre comment utiliser `StartLiveTail`.

SDK pour Kotlin

Joignez les fichiers requis.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Démarrez la session Live Tail.

```
val client = CloudWatchLogsClient.fromEnvironment()

val request = StartLiveTailRequest {
    logGroupIdentifiers = logGroupIdentifiersVal
    logStreamNames = logStreamNamesVal
    logEventFilterPattern = logEventFilterPatternVal
}

val startTime = System.currentTimeMillis()

try {
    client.startLiveTail(request) { response ->
        val stream = response.responseStream
        if (stream != null) {
            /* Set a timeout to unsubscribe from the flow. This will:
            * 1). Close the stream
            * 2). Stop the Live Tail session
            */
            stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                if (value is StartLiveTailResponseStream.SessionStart) {
                    println(value.asSessionStart())
                } else if (value is StartLiveTailResponseStream.SessionUpdate) {
                    for (e in value.asSessionUpdate().sessionResults!!) {
                        println(e)
                    }
                } else {
```



```
                throw IllegalArgumentException("Unknown event type")
            }
        }
    } else {
        throw IllegalArgumentException("No response stream")
    }
}
} catch (e: Exception) {
    println("Exception occurred during StartLiveTail: $e")
    System.exit(1)
}
```

- Pour plus de détails sur l'API, reportez-vous [StartLiveTail](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples de fournisseurs d'identité Amazon Cognito utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec le fournisseur 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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

AdminGetUser

L'exemple de code suivant montre comment utiliser `AdminGetUser`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }


    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [AdminGetUser](#) à la section AWS SDK pour la référence de l'API Kotlin.

AdminInitiateAuth

L'exemple de code suivant montre comment utiliser `AdminInitiateAuth`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [AdminInitiateAuth](#) à la section AWS SDK pour la référence de l'API Kotlin.

AdminRespondToAuthChallenge

L'exemple de code suivant montre comment utiliser `AdminRespondToAuthChallenge`.

SDK pour Kotlin

 Note

Il y en a plus 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.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val respondToAuthChallengeResult =
            identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
        ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [AdminRespondToAuthChallenge](#) à la section AWS SDK pour la référence de l'API Kotlin.

AssociateSoftwareToken

L'exemple de code suivant montre comment utiliser `AssociateSoftwareToken`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
{ identityProviderClient ->
    val tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest)
    val secretCode = tokenResponse.secretCode
    println("Enter this token into Google Authenticator")
    println(secretCode)
    return tokenResponse.session
}
}
```

- Pour plus de détails sur l'API, reportez-vous [AssociateSoftwareToken](#) à la section AWS SDK pour la référence de l'API Kotlin.

ConfirmSignUp

L'exemple de code suivant montre comment utiliser `ConfirmSignUp`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal was confirmed")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ConfirmSignUp](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListUsers

L'exemple de code suivant montre comment utiliser `ListUsers`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListUsers](#) à la section AWS SDK pour la référence de l'API Kotlin.

ResendConfirmationCode

L'exemple de code suivant montre comment utiliser `ResendConfirmationCode`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
```

```
        userNameVal: String?,
    ) {
        val codeRequest =
            ResendConfirmationCodeRequest {
                clientId = clientIdVal
                username = userNameVal
            }

        CognitoIdentityProviderClient { region = "us-east-1" }.use
        { identityProviderClient ->
            val response = identityProviderClient.resendConfirmationCode(codeRequest)
            println("Method of delivery is " +
                (response.codeDeliveryDetails?.deliveryMedium))
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ResendConfirmationCode](#) à la section AWS SDK pour la référence de l'API Kotlin.

SignUp

L'exemple de code suivant montre comment utiliser SignUp.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
        }
}
```



```
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SignUp](#) à la section AWS SDK pour la référence de l'API Kotlin.

VerifySoftwareToken

L'exemple de code suivant montre comment utiliser `VerifySoftwareToken`.

SDK pour Kotlin

Note

Il y en a plus 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.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
```

```
val tokenRequest =
    VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
{ identityProviderClient ->
    val verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest)
    println("The status of the token is ${verifyResponse.status}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [VerifySoftwareToken](#) à la section AWS SDK pour la référence de l'API Kotlin.

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 Kotlin

Note

Il y en a plus 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 Kotlin code example, set up your development environment, including your credentials.

For more information, see the following documentation:

<https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 `initiateAuth` 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.
- */

```
suspend fun main(args: Array<String>) {
    val 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.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]
```

```
// Use the console to get data from the user.
println("**** Enter your use name")
val in0b = Scanner(System.`in`)
val userName = in0b.nextLine()
println(userName)

println("**** Enter your password")
val password: String = in0b.nextLine()

println("**** Enter your email")
val email = in0b.nextLine()

println("**** Signing up $userName")
signUp(clientId, userName, password, email)

println("**** Getting $userName in the user pool")
getAdminUser(userName, poolId)

println("**** Conformation code sent to $userName. Would you like to send a new
code? (Yes/No)")
val ans = in0b.nextLine()

if (ans.compareTo("Yes") == 0) {
    println("**** Sending a new confirmation code")
    resendConfirmationCode(clientId, userName)
}
println("**** Enter the confirmation code that was emailed")
val code = in0b.nextLine()
confirmSignUp(clientId, code, userName)

println("**** Rechecking the status of $userName in the user pool")
getAdminUser(userName, poolId)

val authResponse = checkAuthMethod(clientId, userName, password, poolId)
val mySession = authResponse.session
val newSession = getSecretForAppMFA(mySession)
println("**** Enter the 6-digit code displayed in Google Authenticator")
val myCode = in0b.nextLine()

// Verify the TOTP and register for MFA.
verifyTOTP(newSession, myCode)
println("**** Re-enter a 6-digit code displayed in Google Authenticator")
val mfaCode: String = in0b.nextLine()
```

```

    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
    }
}

```

```

        println("Method of delivery is " +
(response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponses0b = mutableMapOf<String, String>()
    challengeResponses0b["USERNAME"] = userName
    challengeResponses0b["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponses0b
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val respondToAuthChallengeResult =
            identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
        ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }
}

```

```
        CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val verifyResponse =
            identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val tokenResponse =
            identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal was confirmed")
    }
}
```

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```



```
}  
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)
 - [VerifySoftwareToken](#)

Exemples d'Amazon Comprehend utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon Comprehend.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Scénarios](#)

Scénarios

Création d'une application de messagerie

L'exemple de code suivant montre comment créer une application de messagerie à l'aide d'Amazon SQS.

SDK pour Kotlin

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

Exemples DynamoDB utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec DynamoDB.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Principes de base](#)

- [Actions](#)
- [Scénarios](#)

Principes de base

Principes de base

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 Kotlin

Note

Il y en a plus 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.

```
suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
```

```
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
        ${response.tableDescription?.tableArn}")
    }
}
```

Créez une fonction d'assistance pour télécharger et extraire l'exemple de fichier JSON.

```
// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }
}
```

```

    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

```

Obtenez un élément d'une table.

```

suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

```

Exemple complet.

```

suspend fun main(args: Array<String>) {
    val usage = ""
    Usage:
    <fileName>

```

Where:

fileName - The path to the moviedata.json you can download from the Amazon DynamoDB Developer Guide.

```
""
```

```
if (args.size != 1) {
    println(usage)
    exitProcess(1)
}
```

```
// Get the moviedata.json from the Amazon DynamoDB Developer Guide.
```

```
val tableName = "Movies"
```

```
val fileName = args[0]
```

```
val partitionAlias = "#a"
```

```
println("Creating an Amazon DynamoDB table named Movies with a key named id and  
a sort key named title.")
```

```
createScenarioTable(tableName, "year")
```

```
loadData(tableName, fileName)
```

```
getMovie(tableName, "year", "1933")
```

```
scanMovies(tableName)
```

```
val count = queryMovieTable(tableName, "year", partitionAlias)
```

```
println("There are $count Movies released in 2013.")
```

```
deleteIssuesTable(tableName)
```

```
}
```

```
suspend fun createScenarioTable(
```

```
    tableNameVal: String,
```

```
    key: String,
```

```
) {
```

```
    val attDef =
```

```
        AttributeDefinition {
```

```
            attributeName = key
```

```
            attributeType = ScalarAttributeType.N
```

```
        }
```

```
    val attDef1 =
```

```
        AttributeDefinition {
```

```
            attributeName = "title"
```

```
            attributeType = ScalarAttributeType.S
```

```
        }
```

```
    val keySchemaVal =
```

```

        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
    ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)

```



```
val iter: Iterator<JsonNode> = rootNode.iterator()
var currentNode: ObjectNode

var t = 0
while (iter.hasNext()) {
    if (t == 50) {
        break
    }

    currentNode = iter.next() as ObjectNode
    val year = currentNode.path("year").asInt()
    val title = currentNode.path("title").asText()
    val info = currentNode.path("info").toString()
    putMovie(tableName, year, title, info)
    t++
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(
```

```
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"
```

```
// Set up mapping of the partition name with the value.
val attrValues = mutableMapOf<String, AttributeValue>()
attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

val request =
    QueryRequest {
        tableName = tableNameVal
        keyConditionExpression = "$partitionAlias = :$partitionKeyName"
        expressionAttributeNames = attrNameAlias
        this.expressionAttributeValues = attrValues
    }

DynamoDbClient { region = "us-east-1" }.use { ddb ->
    val response = ddb.query(request)
    return response.count
}

suspend fun scanMovies(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la AWS Référence de l'API de SDK pour Kotlin.
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)

- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Interrogation](#)
- [Analyser](#)
- [UpdateItem](#)

Actions

CreateTable

L'exemple de code suivant montre comment utiliser `CreateTable`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createNewTable(
    tableNameVal: String,
    key: String,
): String? {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val provisionedVal =
        ProvisionedThroughput {
```

```
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTable](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteItem

L'exemple de code suivant montre comment utiliser `DeleteItem`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteDynamoDBItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        DeleteItemRequest {
            tableName = tableNameVal
            key = keyToGet
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteItem](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteTable

L'exemple de code suivant montre comment utiliser `DeleteTable`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }
}
```

```
DynamoDbClient { region = "us-east-1" }.use { ddb ->
    ddb.deleteTable(request)
    println("$tableNameVal was deleted")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTable](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetItem

L'exemple de code suivant montre comment utiliser `GetItem`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getSpecificItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
```

```
        println(key1.key)
        println(key1.value)
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetItem](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListTables

L'exemple de code suivant montre comment utiliser `ListTables`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
suspend fun listAllTables() {
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTables](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutItem

L'exemple de code suivant montre comment utiliser `PutItem`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
    songTitle: String,
    songTitleVal: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutItem](#) à la section AWS SDK pour la référence de l'API Kotlin.

Query

L'exemple de code suivant montre comment utiliser `Query`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- Pour plus d'informations sur l'API, consultez [Requête](#) dans la référence d'API du kit AWS SDK pour Kotlin.

Scan

L'exemple de code suivant montre comment utiliser `Scan`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Pour plus d'informations sur l'API, consultez [Analyser](#) dans la référence d'API du kit AWS SDK pour Kotlin.

UpdateItem

L'exemple de code suivant montre comment utiliser `UpdateItem`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
    name: String,
    updateVal: String,
) {
    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] =
        AttributeValueUpdate {
            value = AttributeValue.S(updateVal)
            action = AttributeAction.Put
        }

    val request =
        UpdateItemRequest {
            tableName = tableNameVal
            key = itemKey
            attributeUpdates = updatedValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateItem](#) à la section AWS SDK pour la référence de l'API Kotlin.

Scénarios

Créer une application pour soumettre des données à une table DynamoDB

L'exemple de code suivant montre comment créer une application qui envoie des données à une table Amazon DynamoDB et vous avertit lorsqu'un utilisateur met à jour la table.

SDK pour Kotlin

Indique comment créer une application Android native qui envoie des données à l'aide de l'API Kotlin Amazon DynamoDB et envoie un message texte à l'aide de l'API Kotlin Amazon SNS.

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

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

SDK pour Kotlin

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

L'exemple de code suivant montre comment créer une application Web qui suit les éléments de travail dans une table Amazon DynamoDB et utilise Amazon Simple Email Service (Amazon SES) pour envoyer des rapports.

SDK pour Kotlin

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

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 Kotlin

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun main() {
    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQLBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named id
and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val provisionedVal =
        ProvisionedThroughput {
```

```
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
    ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,
    'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListof<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListof<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
```



```
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

// Set data for Movie 3.
val parametersMovie3 = mutableListOf<AttributeValue>()
parametersMovie3.add(AttributeValue.N("2022"))
parametersMovie3.add(AttributeValue.S("My Movie 3"))
parametersMovie3.add(AttributeValue.S("No Information"))

val statementRequestMovie3 =
    BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersMovie3
    }

// Add all three movies to the list.
val myBatchStatementList = mutableListOf<BatchStatementRequest>()
myBatchStatementList.add(statementRequestMovie1)
myBatchStatementList.add(statementRequestMovie2)
myBatchStatementList.add(statementRequestMovie3)

val batchRequest =
    BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }
val response = ddb.batchExecuteStatement(batchRequest)
println("ExecuteStatement successful: " + response.toString())
println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
        \"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }
}
```

```
// Update record 2.
val parametersRec2 = mutableListOf<AttributeValue>()
parametersRec2.add(AttributeValue.N("2022"))
parametersRec2.add(AttributeValue.S("My Movie 2"))
val statementRequestRec2 =
    BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec2
    }

// Update record 3.
val parametersRec3 = mutableListOf<AttributeValue>()
parametersRec3.add(AttributeValue.N("2022"))
parametersRec3.add(AttributeValue.S("My Movie 3"))
val statementRequestRec3 =
    BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec3
    }

// Add all three movies to the list.
val myBatchStatementList = mutableListOf<BatchStatementRequest>()
myBatchStatementList.add(statementRequestRec1)
myBatchStatementList.add(statementRequestRec2)
myBatchStatementList.add(statementRequestRec3)

val batchRequest =
    BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }

val response = ddb.batchExecuteStatement(batchRequest)
println("ExecuteStatement successful: $response")
println("Updated three movies using a batch command.")
println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
}
```

```
val statementRequestRec1 =
    BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec1
    }

// Specify record 2.
val parametersRec2 = mutableListOf<AttributeValue>()
parametersRec2.add(AttributeValue.N("2022"))
parametersRec2.add(AttributeValue.S("My Movie 2"))
val statementRequestRec2 =
    BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec2
    }

// Specify record 3.
val parametersRec3 = mutableListOf<AttributeValue>()
parametersRec3.add(AttributeValue.N("2022"))
parametersRec3.add(AttributeValue.S("My Movie 3"))
val statementRequestRec3 =
    BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec3
    }

// Add all three movies to the list.
val myBatchStatementList = mutableListOf<BatchStatementRequest>()
myBatchStatementList.add(statementRequestRec1)
myBatchStatementList.add(statementRequestRec2)
myBatchStatementList.add(statementRequestRec3)

val batchRequest =
    BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }

ddb.batchExecuteStatement(batchRequest)
println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
```

```

DeleteTableRequest {
    tableName = tableNameVal
}

DynamoDbClient { region = "us-east-1" }.use { ddb ->
    ddb.deleteTable(request)
    println("$tableNameVal was deleted")
}
}

```

- Pour plus de détails sur l'API, reportez-vous [BatchExecuteStatement](#) à la section AWS SDK pour la référence de l'API Kotlin.

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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun main(args: Array<String>) {
    val usage = ""
    Usage:
        <fileName>

    Where:
        fileName - The path to the moviedata.json file You can download from the
        Amazon DynamoDB Developer Guide.

```

```

    ""

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQ"

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val fileName = args[0]
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key named
id and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("***** Getting data from the MoviesPartiQ table.")
    getMoviePartiQL(ddb)

    println("***** Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("***** Updating a record.")
    updateTableItemPartiQL(ddb)

    println("***** Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("***** Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}

suspend fun createTablePartiQL(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }
}

```

```
val attDef1 =
    AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

val keySchemaVal =
    KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

val keySchemaVal1 =
    KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

val provisionedVal =
    ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

val request =
    CreateTableRequest {
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

val response = ddb.createTable(request)
ddb.waitUntilTableExists {
    // suspend call
    tableName = tableNameVal
}
println("The table was successfully created
${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(
    ddb: DynamoDbClient,
    fileName: String,
```

```

) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {
        if (t == 200) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()

```

```
parameters.add(AttributeValue.N("2020"))
parameters.add(AttributeValue.S("My Movie"))
parameters.add(AttributeValue.S("No Info"))
executeStatementPartiQL(ddb, sqlStatement, parameters)
println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
    Cooper\", \"Ernest B. Schoedsack\" where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>,
): ExecuteStatementResponse {
    val request =
```



```
ExecuteStatementRequest {
    statement = statementVal
    parameters = parametersVal
}

return ddb.executeStatement(request)
}
```

- Pour plus de détails sur l'API, reportez-vous [ExecuteStatement](#) à la section AWS SDK pour la référence de l'API Kotlin.

EC2 Exemples Amazon utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon. EC2

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Amazon EC2

Les exemples de code suivants montrent comment commencer à utiliser Amazon EC2.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
                ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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 Kotlin

 Note

Il y en a plus 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 Kotlin 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-kotlin/latest/developer-guide/setup.html

This Kotlin 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.
16. Deletes the security group.
17. Deletes the key pair.
*/

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
```

```

val usage = """
  Usage:
    <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

  Where:
    keyName - A key pair name (for example, TestKeyPair).
    fileName - A file name where the key information is written to.
    groupName - The name of the security group.
    groupDesc - The description of the security group.
    vpcId - A VPC ID. You can get this value from the AWS Management
Console.
    myIpAddress - The IP address of your development machine.

"""

if (args.size != 6) {
    println(usage)
    exitProcess(0)
}

val keyName = args[0]
val fileName = args[1]
val groupName = args[2]
val groupDesc = args[3]
val vpcId = args[4]
val myIpAddress = args[5]
var newInstanceId: String? = ""

println(DASHES)
println("Welcome to the Amazon EC2 example scenario.")
println(DASHES)

println(DASHES)
println("1. Create an RSA key pair and save the private key material as a .pem
file.")
createKeyPairSc(keyName, fileName)
println(DASHES)

println(DASHES)
println("2. List key pairs.")
describeEC2KeysSc()
println(DASHES)

println(DASHES)

```

```
println("3. Create a security group.")
val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId, myIpAddress)
println(DASHES)

println(DASHES)
println("4. Display security group info for the newly created security group.")
describeSecurityGroupsSc(groupId.toString())
println(DASHES)

println(DASHES)
println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in the
name.")
val instanceId = getParaValuesSc()
if (instanceId == "") {
    println("The instance Id value isn't valid.")
    exitProcess(0)
}
println("The instance Id is $instanceId.")
println(DASHES)

println(DASHES)
println("6. Get more information about an amzn2 image and return the AMI
value.")
val amiValue = instanceId?.let { describeImageSc(it) }
if (instanceId == "") {
    println("The instance Id value is invalid.")
    exitProcess(0)
}
println("The AMI value is $amiValue.")
println(DASHES)

println(DASHES)
println("7. Get a list of instance types.")
val instanceType = getInstanceTypesSc()
println(DASHES)

println(DASHES)
println("8. Create an instance.")
if (amiValue != null) {
    newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
    println("The instance Id is $newInstanceId")
}
println(DASHES)
```

```
println(DASHES)
println("9. Display information about the running instance. ")
var ipAddress = describeEC2InstancesSc(newInstanceId)
println("You can SSH to the instance using this command:")
println("ssh -i " + fileName + "ec2-user@" + ipAddress)
println(DASHES)

println(DASHES)
println("10. Stop the instance.")
if (newInstanceId != null) {
    stopInstanceSc(newInstanceId)
}
println(DASHES)

println(DASHES)
println("11. Start the instance.")
if (newInstanceId != null) {
    startInstanceSc(newInstanceId)
}
ipAddress = describeEC2InstancesSc(newInstanceId)
println("You can SSH to the instance using this command:")
println("ssh -i " + fileName + "ec2-user@" + ipAddress)
println(DASHES)

println(DASHES)
println("12. Allocate an Elastic IP address and associate it with the
instance.")
val allocationId = allocateAddressSc()
println("The allocation Id value is $allocationId")
val associationId = associateAddressSc(newInstanceId, allocationId)
println("The associate Id value is $associationId")
println(DASHES)

println(DASHES)
println("13. Describe the instance again.")
ipAddress = describeEC2InstancesSc(newInstanceId)
println("You can SSH to the instance using this command:")
println("ssh -i " + fileName + "ec2-user@" + ipAddress)
println(DASHES)

println(DASHES)
println("14. Disassociate and release the Elastic IP address.")
disassociateAddressSc(associationId)
releaseEC2AddressSc(allocationId)
```

```
println(DASHES)

println(DASHES)
println("15. Terminate the instance and use a waiter.")
if (newInstanceId != null) {
    terminateEC2Sc(newInstanceId)
}
println(DASHES)

println(DASHES)
println("16. Delete the security group.")
if (groupId != null) {
    deleteEC2SecGroupSc(groupId)
}
println(DASHES)

println(DASHES)
println("17. Delete the key pair.")
deleteKeysSc(keyName)
println(DASHES)

println(DASHES)
println("You successfully completed the Amazon EC2 scenario.")
println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
    }
}
```

```
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitForInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
```



```
        instanceIdVal: String?,
        allocationIdVal: String?,
    ): String? {
        val associateRequest =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        Ec2Client { region = "us-west-2" }.use { ec2 ->
            val associateResponse = ec2.associateAddress(associateRequest)
            return associateResponse.associationId
        }
    }

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
minutes.")
        ec2.waitForInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

```

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
                    ?.get(0)
                    ?.instances
                    ?.get(0)
                    ?.state
                    ?.name
                    ?.value

            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
                }
            }
        }
    }
}

```

```

        println("Instance state is
${response.reservations!!.get(0).instances?.get(0)?.state}")
        pubAddress =
            response.reservations!!
                .get(0)
                .instances
                ?.get(0)
                ?.publicIpAddress
                .toString()
        println("Instance address is $pubAddress")
        isRunning = true
    }
}
}
}
return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
            imageId = amiIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
$amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.

```

```
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
                ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
                ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
            ${response.images?.get(0)?.description}")
        println("The name of the first image is ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
}
```

```
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() + "
and group VPC " + group.vpcId)
        }
    }
}
}
```

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

```
    }
}

suspend fun describeEC2KeysSc() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
                ${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)

- [DeleteSecurityGroup](#)
- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

Actions

AllocateAddress

L'exemple de code suivant montre comment utiliser `AllocateAddress`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
    }
}
```



```
    val allocationIdVal = allocateResponse.allocationId

    val request =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    val associateResponse = ec2.associateAddress(request)
    return associateResponse.associationId
}
}
```

- Pour plus de détails sur l'API, reportez-vous [AssociateAddress](#) à la section AWS SDK pour la référence de l'API Kotlin.

AssociateAddress

L'exemple de code suivant montre comment utiliser `AssociateAddress`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
    }
}
```

```
        return associateResponse.associationId
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [AssociateAddress](#) à la section AWS SDK pour la référence de l'API Kotlin.

AuthorizeSecurityGroupIngress

L'exemple de code suivant montre comment utiliser `AuthorizeSecurityGroupIngress`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
```

```
        IpPermission {
            ipProtocol = "tcp"
            toPort = 80
            fromPort = 80
            ipRanges = listOf(ipRange)
        }

    val ipPerm2 =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 22
            fromPort = 22
            ipRanges = listOf(ipRange)
        }

    val authRequest =
        AuthorizeSecurityGroupIngressRequest {
            groupName = groupNameVal
            ipPermissions = listOf(ipPerm, ipPerm2)
        }
    ec2.authorizeSecurityGroupIngress(authRequest)
    println("Successfully added ingress policy to Security Group $groupNameVal")
    return resp.groupId
}
}
```

- Pour plus de détails sur l'API, reportez-vous [AuthorizeSecurityGroupIngress](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateKeyPair

L'exemple de code suivant montre comment utiliser `CreateKeyPair`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyPair](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateSecurityGroup

L'exemple de code suivant montre comment utiliser `CreateSecurityGroup`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
```

```
    val resp = ec2.createSecurityGroup(request)
    val ipRange =
        IpRange {
            cidrIp = "0.0.0.0/0"
        }

    val ipPerm =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 80
            fromPort = 80
            ipRanges = listOf(ipRange)
        }

    val ipPerm2 =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 22
            fromPort = 22
            ipRanges = listOf(ipRange)
        }

    val authRequest =
        AuthorizeSecurityGroupIngressRequest {
            groupName = groupNameVal
            ipPermissions = listOf(ipPerm, ipPerm2)
        }
    ec2.authorizeSecurityGroupIngress(authRequest)
    println("Successfully added ingress policy to Security Group $groupNameVal")
    return resp.groupId
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateSecurityGroup](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteKeyPair

L'exemple de code suivant montre comment utiliser `DeleteKeyPair`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyPair](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteSecurityGroup

L'exemple de code suivant montre comment utiliser DeleteSecurityGroup.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
}
```

```
    }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteSecurityGroup](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeInstanceTypes

L'exemple de code suivant montre comment utiliser `DescribeInstanceTypes`.

SDK pour Kotlin

Note

Il y en a plus 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.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
```

```

        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
                ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
                ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstanceTypes](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeInstances

L'exemple de code suivant montre comment utiliser `DescribeInstances`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstances(request)
        response.reservations?.forEach { reservation ->
            reservation.instances?.forEach { instance ->
                println("Instance Id is ${instance.instanceId}")
                println("Image id is ${instance.imageId}")
                println("Instance type is ${instance.instanceType}")
            }
        }
    }
}

```



```

        println("Instance state name is ${instance.state?.name}")
        println("monitoring information is ${instance.monitoring?.state}")
    }
}
}
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeKeyPairs

L'exemple de code suivant montre comment utiliser `DescribeKeyPairs`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun describeEC2Keys() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
            ${ keyPair.keyFingerprint}")
        }
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeKeyPairs](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeSecurityGroups

L'exemple de code suivant montre comment utiliser `DescribeSecurityGroups`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->


        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
                ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSecurityGroups](#) à la section AWS SDK pour la référence de l'API Kotlin.

DisassociateAddress

L'exemple de code suivant montre comment utiliser `DisassociateAddress`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DisassociateAddress](#) à la section AWS SDK pour la référence de l'API Kotlin.

ReleaseAddress

L'exemple de code suivant montre comment utiliser `ReleaseAddress`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ReleaseAddress](#) à la section AWS SDK pour la référence de l'API Kotlin.

RunInstances

L'exemple de code suivant montre comment utiliser `RunInstances`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
                value = name
            }

        val requestTags =
            CreateTagsRequest {
                resources = listOf(instanceId.toString())
                tags = listOf(tag)
            }
        ec2.createTags(requestTags)
    }
```

```
        println("Successfully started EC2 Instance $instanceId based on AMI $amiId")
        return instanceId
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [RunInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

StartInstances

L'exemple de code suivant montre comment utiliser `StartInstances`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StartInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

StopInstances

L'exemple de code suivant montre comment utiliser `StopInstances`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }


    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
minutes.")
        ec2.waitForInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StopInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

TerminateInstances

L'exemple de code suivant montre comment utiliser `TerminateInstances`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is ${instance.instanceId}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [TerminateInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples Amazon ECR utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon ECR.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Amazon ECR

Les exemples de code suivants montrent comment commencer à utiliser Amazon ECR.

SDK pour Kotlin

Note

Il y en a plus 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 aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
        Usage: <repositoryName>

        Where:
            repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }
}
```



```
EcrClient { region = "us-east-1" }.use { ecrClient ->
    val imageResponse = ecrClient.listImages(listImages)
    imageResponse.imageIds?.forEach { imageId ->
        println("Image tag: ${imageId.imageTag}")
    }
}
```

- Pour plus de détails sur l'API, voir [ListImages dans](#) le AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)
- [Actions](#)


Principes de base

Principes de base

L'exemple de code suivant illustre comment :

- Créez un référentiel Amazon ECR.
- Définissez les politiques du référentiel.
- Récupérez le référentiel URIs.
- Obtenez des jetons d'autorisation Amazon ECR.
- Définissez des politiques de cycle de vie pour les référentiels Amazon ECR.
- Transférez une image Docker vers un référentiel Amazon ECR.
- Vérifiez l'existence d'une image dans un référentiel Amazon ECR.
- Répertoirez les référentiels Amazon ECR pour votre compte et obtenez des informations les concernant.
- Supprimez les référentiels Amazon ECR.

SDK pour Kotlin

 Note

Il y en a plus 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 d'Amazon ECR.

```
import java.util.Scanner

/**
 * Before running this Kotlin 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-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with the
 * Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a local
 * image,
 * this program will not successfully run. For more information including how to
 * create the local
 * image, see:
 *
 * /getting_started_scenarios/ecr_scenario/README
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage: <iamRoleARN> <accountId>
    """
}
```

Where:

iamRoleARN - The IAM role ARN that has the necessary permissions to access and manage the Amazon ECR repository.
 accountId - Your AWS account number.

```

    """.trimIndent()

// if (args.size != 2) {
//     println(usage)
//     return
// }

var iamRole = "arn:aws:iam::814548047983:role/Admin"
var localImageName: String
var accountId = "814548047983"
val ecrActions = ECRActions()
val scanner = Scanner(System.`in`)

println(
    """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
    container registry
        service provided by AWS. It allows developers and organizations to securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images throughout
    their lifecycle,
        from building and testing to production deployment.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
    provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
    developers to
        automate the storage, retrieval, and management of container images as part
    of their application
        deployment pipelines. With ECR, teams can focus on building and deploying
    their
        applications without having to worry about the underlying infrastructure
    required to
        host and manage a container registry.

        This scenario walks you through how to perform key operations for this
    service.
        Let's get started...
    """
)

```

```
    You have two choices:
        1 - Run the entire program.
        2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
        this program that did not complete).
```

```
        """.trimIndent(),
    )

while (true) {
    val input = scanner.nextLine()
    if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
        println("Continuing with the program...")
        println("")
        break
    } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
        val repoName = "echo-text"
        ecrActions.deleteECRRepository(repoName)
        return
    } else {
        // Handle invalid input.
        println("Invalid input. Please try again.")
    }
}
}
```

```
waitForInputToContinue(scanner)
println(DASHES)
println(
    """
    1. Create an ECR repository.
```

The first task is to ensure we have a local Docker image named echo-text. If this image exists, then an Amazon ECR repository is created.

An ECR repository is a private Docker container repository provided by Amazon Web Services (AWS). It is a managed service that makes it easy to store, manage, and deploy Docker container images.

```
        """.trimIndent(),
    )

// Ensure that a local docker image named echo-text exists.
val doesExist = ecrActions.listLocalImages()
```

```

val repoName: String
if (!doesExist) {
    println("The local image named echo-text does not exist")
    return
} else {
    localImageName = "echo-text"
    repoName = "echo-text"
}

val repoArn = ecrActions.createECRRepository(repoName).toString()
println("The ARN of the ECR repository is $repoArn")
waitForInputToContinue(scanner)

```

```

println(DASHES)
println(
    """
    2. Set an ECR repository policy.

```

Setting an ECR repository policy using the `setRepositoryPolicy` function is crucial for maintaining the security and integrity of your container images. The repository policy allows you to define specific rules and restrictions for accessing and managing the images stored within your ECR repository.

```

    """.trimIndent(),
)
waitForInputToContinue(scanner)
ecrActions.setRepoPolicy(repoName, iamRole)
waitForInputToContinue(scanner)

```

```

println(DASHES)
println(
    """
    3. Display ECR repository policy.

```

Now we will retrieve the ECR policy to ensure it was successfully set.

```

    """.trimIndent(),
)
waitForInputToContinue(scanner)
val policyText = ecrActions.getRepoPolicy(repoName)
println("Policy Text:")
println(policyText)

```

```
waitForInputToContinue(scanner)
```

```
println(DASHES)
```

```
println(
```

```
    ""
```

```
    4. Retrieve an ECR authorization token.
```

You need an authorization token to securely access and interact with the Amazon ECR registry.

The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible for securely accessing and interacting with an Amazon ECR repository. This operation is responsible for obtaining a valid authorization token, which is required to authenticate your requests to the ECR service.

Without a valid authorization token, you would not be able to perform any operations on the ECR repository, such as pushing, pulling, or managing your Docker images.

```
    ""}.trimIndent(),
```

```
)
```

```
waitForInputToContinue(scanner)
```

```
ecrActions.getAuthToken()
```

```
waitForInputToContinue(scanner)
```

```
println(DASHES)
```

```
println(
```

```
    ""
```

```
    5. Get the ECR Repository URI.
```

The URI of an Amazon ECR repository is important. When you want to deploy a container image to a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)

or Amazon Elastic Container Service (ECS), you need to specify the full image URI, which includes the ECR repository URI. This allows the container runtime to pull the correct container image from the ECR repository.

```
    ""}.trimIndent(),
```

```
)
```

```
waitForInputToContinue(scanner)
val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
println("The repository URI is $repositoryURI")
waitForInputToContinue(scanner)
```

```
println(DASHES)
```

```
println(
```

```
    ""
```

```
    6. Set an ECR Lifecycle Policy.
```

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored in your ECR repositories.

These policies allow you to automatically remove old or unused Docker images from your repositories, freeing up storage space and reducing costs.

```
    "").trimIndent(),
```

```
)
```

```
waitForInputToContinue(scanner)
```

```
val pol = ecrActions.setLifecyclePolicy(repoName)
```

```
println(pol)
```

```
waitForInputToContinue(scanner)
```

```
println(DASHES)
```

```
println(
```

```
    ""
```

```
    7. Push a docker image to the Amazon ECR Repository.
```

The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR repository.

It sets up the Docker client by connecting to the local Docker host using the default port.

It then retrieves the authorization token for the ECR repository by making a call to the AWS SDK.

The method uses the authorization token to create an `AuthConfig` object, which is used to authenticate

the Docker client when pushing the image. Finally, the method tags the Docker image with the specified

repository name and image tag, and then pushes the image to the ECR repository using the Docker client.

If the push operation is successful, the method prints a message indicating that the image was pushed to ECR.

```
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("9. As an optional step, you can interact with the image in Amazon ECR
by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
image? (y/n)")
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
you need to authenticate with the registry. You can do this using the AWS CLI:

                aws ecr get-login-password --region us-east-1 | docker login --username
AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

                aws ecr describe-images --repository-name $repoName --image-ids imageTag=
$localImageName

            3. Run the Docker container and view the output using this command:

                docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
            """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
```



```

        """
        If the repository isn't empty, you must either delete the contents of the
        repository
        or use the force option (used in this scenario) to delete the repository and
        have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }
}
}

```

Une classe wrapper pour les méthodes du SDK Amazon ECR.

```

import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest

```

```
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
            default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =
                NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
                DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }
}

/**
 * Sets the lifecycle policy for the specified repository.
 *
 * @param repoName the name of the repository for which to set the lifecycle
 * policy.
 */
```

```

suspend fun setLifecyclePolicy(repoName: String): String? {
    val polText =
        """
        {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14
                    },
                    "action": {
                        "type": "expire"
                    }
                }
            ]
        }

        """.trimIndent()
    val lifecyclePolicyPreviewRequest =
        StartLifecyclePolicyPreviewRequest {
            lifecyclePolicyText = polText
            repositoryName = repoName
        }

    // Execute the request asynchronously.
    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val response =
            ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
        return response.lifecyclePolicyText
    }
}

/**
 * Retrieves the repository URI for the specified repository name.
 *
 * @param repoName the name of the repository to retrieve the URI for.
 * @return the repository URI for the specified repository name.
 */
suspend fun getRepositoryURI(repoName: String?): String? {

```

```
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
                ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
                describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }

    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
     * (ECR).
     */
    suspend fun getAuthToken() {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }

    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
```

```

suspend fun getRepoPolicy(repoName: String?): String? {
    require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

    // Create the request
    val getRepositoryPolicyRequest =
        GetRepositoryPolicyRequest {
            repositoryName = repoName
        }
    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
        val responseText = response.policyText
        return responseText
    }
}

/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
    val policyDocumentTemplate =
        """
        {
            "Version" : "2012-10-17",
            "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                    "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
            } ]
        }
        """
    val setRepositoryPolicyRequest =
        SetRepositoryPolicyRequest {

```

```

        repositoryName = repoName
        policyText = policyDocumentTemplate
    }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
        if (response != null) {
            println("Repository policy set successfully.")
        }
    }
}

/**
 * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
 *
 * @param repoName the name of the repository to create.
 * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
 * @throws RepositoryAlreadyExistsException if the repository exists.
 * @throws EcrException if an error occurs while creating the
repository.
 */
suspend fun createECRRepository(repoName: String?): String? {
    val request =
        CreateRepositoryRequest {
            repositoryName = repoName
        }

    return try {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.createRepository(request)
            response.repository?.repositoryArn
        }
    } catch (e: RepositoryAlreadyExistsException) {
        println("Repository already exists: $repoName")
        repoName?.let { getRepoARN(it) }
    } catch (e: EcrException) {
        println("An error occurred: ${e.message}")
        null
    }
}

suspend fun getRepoARN(repoName: String): String? {

```

```

    // Fetch the existing repository's ARN.
    val describeRequest =
        DescribeRepositoriesRequest {
            repositoryNames = listOf(repoName)
        }
    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val describeResponse = ecrClient.describeRepositories(describeRequest)
        return describeResponse.repositories?.get(0)?.repositoryArn
    }
}

fun listLocalImages(): Boolean = try {
    val images = getDockerClient()?.listImagesCmd()?.exec()
    images?.any { image ->
        image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
    } ?: false
} catch (ex: Exception) {
    println("ERROR: ${ex.message}")
    false
}

/**
 * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
 *
 * @param repoName the name of the ECR repository to push the image to.
 * @param imageName the name of the Docker image.
 */
suspend fun pushDockerImage(
    repoName: String,
    imageName: String,
) {
    println("Pushing $imageName to $repoName will take a few seconds")
    val authConfig = getAuthConfig(repoName)

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val desRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        val describeRepoResponse = ecrClient.describeRepositories(desRequest)
        val repoData =

```

```

        describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
            ?: throw RuntimeException("Repository not found: $repoName")

        val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
        if (tagImageCmd != null) {
            tagImageCmd.exec()
        }
        val pushImageCmd =
            repoData.repositoryUri?.let {
                dockerClient?.pushImageCmd(it)
                    // ?.withTag("latest")
                    ?.withAuthConfig(authConfig)
            }

        try {
            if (pushImageCmd != null) {
                pushImageCmd.start().awaitCompletion()
            }
            println("The $imageName was pushed to Amazon ECR")
        } catch (e: IOException) {
            throw RuntimeException(e)
        }
    }
}

/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag       The tag of the image to verify.
 */
suspend fun verifyImage(
    repoName: String?,
    imageTagVal: String?,
) {
    require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
    require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
be null or empty" }
}

```



```

    val imageId =
        ImageIdentifier {
            imageTag = imageTagVal
        }
    val request =
        DescribeImagesRequest {
            repositoryName = repoName
            imageIds = listOf(imageId)
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val describeImagesResponse = ecrClient.describeImages(request)
        if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
            println("Image is present in the repository.")
        } else {
            println("Image is not present in the repository.")
        }
    }
}

/**
 * Deletes an ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 */
suspend fun deleteECRRepository(repoName: String) {
    if (repoName.isNullOrEmpty()) {
        throw IllegalArgumentException("Repository name cannot be null or
empty")
    }

    val repositoryRequest =
        DeleteRepositoryRequest {
            force = true
            repositoryName = repoName
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        ecrClient.deleteRepository(repositoryRequest)
        println("You have successfully deleted the $repoName repository")
    }
}

```

```
// Return an AuthConfig.
private suspend fun getAuthConfig(repoName: String): AuthConfig {
    EcrClient { region = "us-east-1" }.use { ecrClient ->
        // Retrieve the authorization token for ECR.
        val response = ecrClient.getAuthorizationToken()
        val authorizationData = response.authorizationData?.get(0)
        val token = authorizationData?.authorizationToken
        val decodedToken = String(Base64.getDecoder().decode(token))
        val password = decodedToken.substring(4)

        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        val descrRepoResponse = ecrClient.describeRepositories(request)
        val repoData = descrRepoResponse.repositories?.firstOrNull
        { it.repositoryName == repoName }
        val registryURL: String = repoData?.repositoryUri?.split("/")?.get(0) ?:
        ""

        return AuthConfig()
            .withUsername("AWS")
            .withPassword(password)
            .withRegistryAddress(registryURL)
    }
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la AWS Référence de l'API de SDK pour Kotlin.
 - [CreateRepository](#)
 - [DeleteRepository](#)
 - [DescribeImages](#)
 - [DescribeRepositories](#)
 - [GetAuthorizationToken](#)
 - [GetRepositoryPolicy](#)
 - [SetRepositoryPolicy](#)

- [StartLifecyclePolicyPreview](#)

Actions

CreateRepository

L'exemple de code suivant montre comment utiliser `CreateRepository`.

SDK pour Kotlin

Note

Il y en a plus 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 Amazon Elastic Container Registry (Amazon ECR) repository.
 *
 * @param repoName the name of the repository to create.
 * @return the Amazon Resource Name (ARN) of the created repository, or an empty
 string if the operation failed.
 * @throws RepositoryAlreadyExistsException if the repository exists.
 * @throws EcrException if an error occurs while creating the
 repository.
 */
suspend fun createECRRepository(repoName: String?): String? {
    val request =
        CreateRepositoryRequest {
            repositoryName = repoName
        }

    return try {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.createRepository(request)
            response.repository?.repositoryArn
        }
    } catch (e: RepositoryAlreadyExistsException) {
        println("Repository already exists: $repoName")
        repoName?.let { getRepoARN(it) }
    }
}
```

```

        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateRepository](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteRepository

L'exemple de code suivant montre comment utiliser `DeleteRepository`.

SDK pour Kotlin

Note

Il y en a plus 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 ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 */
suspend fun deleteECRRepository(repoName: String) {
    if (repoName.isNullOrEmpty()) {
        throw IllegalArgumentException("Repository name cannot be null or
empty")
    }

    val repositoryRequest =
        DeleteRepositoryRequest {
            force = true
            repositoryName = repoName
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->

```

```
        ecrClient.deleteRepository(repositoryRequest)
        println("You have successfully deleted the $repoName repository")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteRepository](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeImages

L'exemple de code suivant montre comment utiliser `DescribeImages`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag       The tag of the image to verify.
 */
suspend fun verifyImage(
    repoName: String?,
    imageTagVal: String?,
) {
    require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
    require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
be null or empty" }

    val imageId =
        ImageIdentifier {
            imageTag = imageTagVal
        }
}
```

```
    }
    val request =
        DescribeImagesRequest {
            repositoryName = repoName
            imageIds = listOf(imageId)
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val describeImagesResponse = ecrClient.describeImages(request)
        if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
            println("Image is present in the repository.")
        } else {
            println("Image is not present in the repository.")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeImages](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeRepositories

L'exemple de code suivant montre comment utiliser `DescribeRepositories`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Retrieves the repository URI for the specified repository name.
 *
 * @param repoName the name of the repository to retrieve the URI for.
 * @return the repository URI for the specified repository name.
 */
```

```

suspend fun getRepositoryURI(repoName: String?): String? {
    require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
    val request =
        DescribeRepositoriesRequest {
            repositoryNames = listOf(repoName)
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
        if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
            return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
        } else {
            println("No repositories found for the given name.")
            return ""
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeRepositories](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetAuthorizationToken

L'exemple de code suivant montre comment utiliser `GetAuthorizationToken`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

/**
 * Retrieves the authorization token for Amazon Elastic Container Registry
 (ECR).

```

```

*
*/
suspend fun getAuthToken() {
    EcrClient { region = "us-east-1" }.use { ecrClient ->
        // Retrieve the authorization token for ECR.
        val response = ecrClient.getAuthorizationToken()
        val authorizationData = response.authorizationData?.get(0)
        val token = authorizationData?.authorizationToken
        if (token != null) {
            println("The token was successfully retrieved.")
        }
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [GetAuthorizationToken](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetRepositoryPolicy

L'exemple de code suivant montre comment utiliser `GetRepositoryPolicy`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

/**
 * Gets the repository policy for the specified repository.
 *
 * @param repoName the name of the repository.
 */
suspend fun getRepoPolicy(repoName: String?): String? {
    require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot be null or empty" }

    // Create the request

```



```

    val getRepositoryPolicyRequest =
        GetRepositoryPolicyRequest {
            repositoryName = repoName
        }
    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
        val responseText = response.policyText
        return responseText
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [GetRepositoryPolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

PushImageCmd

L'exemple de code suivant montre comment utiliser `PushImageCmd`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

/**
 * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
 * repository.
 *
 * @param repoName the name of the ECR repository to push the image to.
 * @param imageName the name of the Docker image.
 */
suspend fun pushDockerImage(
    repoName: String,
    imageName: String,
) {
    println("Pushing $imageName to $repoName will take a few seconds")
    val authConfig = getAuthConfig(repoName)

```

```

EcrClient { region = "us-east-1" }.use { ecrClient ->
    val desRequest =
        DescribeRepositoriesRequest {
            repositoryNames = listOf(repoName)
        }

    val describeRepoResponse = ecrClient.describeRepositories(desRequest)
    val repoData =
        describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
            ?: throw RuntimeException("Repository not found: $repoName")

    val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
    if (tagImageCmd != null) {
        tagImageCmd.exec()
    }
    val pushImageCmd =
        repoData.repositoryUri?.let {
            dockerClient?.pushImageCmd(it)
                // ?.withTag("latest")
                ?.withAuthConfig(authConfig)
        }

    try {
        if (pushImageCmd != null) {
            pushImageCmd.start().awaitCompletion()
        }
        println("The $imageName was pushed to Amazon ECR")
    } catch (e: IOException) {
        throw RuntimeException(e)
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [PushImageCmd](#) à la section AWS SDK pour la référence de l'API Kotlin.

SetRepositoryPolicy

L'exemple de code suivant montre comment utiliser `SetRepositoryPolicy`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
    val policyDocumentTemplate =
        """
        {
            "Version" : "2012-10-17",
            "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                    "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
            } ]
        }
        """.trimIndent()
    val setRepositoryPolicyRequest =
        SetRepositoryPolicyRequest {
            repositoryName = repoName
            policyText = policyDocumentTemplate
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
    }
}
```

```
        if (response != null) {
            println("Repository policy set successfully.")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SetRepositoryPolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

StartLifecyclePolicyPreview

L'exemple de code suivant montre comment utiliser `StartLifecyclePolicyPreview`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag       The tag of the image to verify.
 */
suspend fun verifyImage(
    repoName: String?,
    imageTagVal: String?,
) {
    require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
    require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
be null or empty" }

    val imageId =
        ImageIdentifier {
```

```
        imageTag = imageTagVal
    }
    val request =
        DescribeImagesRequest {
            repositoryName = repoName
            imageIds = listOf(imageId)
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val describeImagesResponse = ecrClient.describeImages(request)
        if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
            println("Image is present in the repository.")
        } else {
            println("Image is not present in the repository.")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StartLifecyclePolicyPreview](#) à la section AWS SDK pour la référence de l'API Kotlin.

OpenSearch Exemples de services utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin with OpenSearch Service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createNewDomain(domainNameVal: String?) {
    val clusterConfig0b =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
                OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
                OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }

    val ebsOptions0b =
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptions0b =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
        }
}
```

```
        clusterConfig = clusterConfig0b
        ebsOptions = ebsOptions0b
        nodeToNodeEncryptionOptions = encryptionOptions0b
    }

    println("Sending domain creation request...")
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateDomain](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteDomain

L'exemple de code suivant montre comment utiliser `DeleteDomain`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteDomain](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListDomainNames

L'exemple de code suivant montre comment utiliser `ListDomainNames`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listAllDomains() {
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
            searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDomainNames](#) à la section AWS SDK pour la référence de l'API Kotlin.

UpdateDomainConfig

L'exemple de code suivant montre comment utiliser `UpdateDomainConfig`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfig0b =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfig0b
        }

    println("Sending domain update request...")
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateDomainConfig](#) à la section AWS SDK pour la référence de l'API Kotlin.

EventBridge exemples d'utilisation du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec. EventBridge

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 démarrer avec EventBridge.

SDK pour Kotlin

Note

Il y en a plus 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 aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request =
        ListEventBusesRequest {
            limit = 10
        }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val response: ListEventBusesResponse = eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListEventBuses](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)

- [Actions](#)

Principes de base

Principes de base

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 Kotlin

Note

Il y en a plus 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 Kotlin 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-kotlin/latest/developer-guide/setup.html
```

```
This Kotlin example performs the following tasks with Amazon EventBridge:
```

1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon EventBridge.
2. Creates an 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 S3 bucket.
11. Disables a specific rule.
12. Checks and prints 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 S3 bucket.
16. Updates the rule to a custom rule pattern.
17. Sends an event to trigger the rule.
18. Cleans up resources.

```
*/
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main(args: Array<String>) {
```

```
    val 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.
```

```
    ""
```

```
    val polJSON =
```

```
        "{" +
```

```
            "\"Version\": \"2012-10-17\", " +
```

```
            "\"Statement\": [{" +
```

```
                "\"Effect\": \"Allow\", " +
```

```
                "\"Principal\": {" +
```

```
                    "\"Service\": \"events.amazonaws.com\" " +
```

```
                }, " +
```

```
                "\"Action\": \"sts:AssumeRole\" " +
```

```
            }]" +
```

```
        }"
```

```
    if (args.size != 4) {
```

```
        println(usage)
```

```
        exitProcess(1)
```

```
}

val sc = Scanner(System.`in`)
val roleName = args[0]
val bucketName = args[1]
val topicName = args[2]
val eventRuleName = args[3]

println(DASHES)
println("Welcome to the Amazon EventBridge example scenario.")
println(DASHES)

println(DASHES)
println("1. Create an AWS Identity and Access Management (IAM) role to use with
Amazon EventBridge.")
val roleArn = createIAMRole(roleName, polJSON)
println(DASHES)

println(DASHES)
println("2. Create an S3 bucket with EventBridge events enabled.")
if (checkBucket(bucketName)) {
    println("$bucketName already exists. Ending this scenario.")
    exitProcess(1)
}

createBucket(bucketName)
delay(3000)
setBucketNotification(bucketName)
println(DASHES)

println(DASHES)
println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
delay(10000)
addEventRule(roleArn, bucketName, eventRuleName)
println(DASHES)

println(DASHES)
println("4. List rules on the event bus.")
listRules()
println(DASHES)

println(DASHES)
```

```
println("5. Create a new SNS topic for testing and let the user subscribe to the
topic.")
val topicArn = createSnsTopic(topicName)
println(DASHES)

println(DASHES)
println("6. Add a target to the rule that sends an email to the specified
topic.")
println("Enter your email to subscribe to the Amazon SNS topic:")
val email = sc.nextLine()
subEmail(topicArn, email)
println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
sc.nextLine()
println(DASHES)

println(DASHES)
println("7. Create an EventBridge event that sends an email when an Amazon S3
object is created.")
addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName, bucketName)
println(DASHES)

println(DASHES)
println("8. List targets.")
listTargets(eventRuleName)
println(DASHES)

println(DASHES)
println(" 9. List the rules for the same target.")
listTargetRules(topicArn)
println(DASHES)

println(DASHES)
println("10. Trigger the rule by uploading a file to the S3 bucket.")
println("Press Enter to continue.")
sc.nextLine()
uploadTextFiletoS3(bucketName)
println(DASHES)

println(DASHES)
println("11. Disable a specific rule.")
changeRuleState(eventRuleName, false)
println(DASHES)
```

```
println(DASHES)
println("12. Check and print the state of the rule.")
checkRule(eventRuleName)
println(DASHES)

println(DASHES)
println("13. Add a transform to the rule to change the text of the email.")
updateSnsEventRule(topicArn, eventRuleName)
println(DASHES)

println(DASHES)
println("14. Enable a specific rule.")
changeRuleState(eventRuleName, true)
println(DASHES)

println(DASHES)
println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
println("Press Enter to continue.")
sc.nextLine()
uploadTextFiletoS3(bucketName)
println(DASHES)

println(DASHES)
println("16. Update the rule to a custom rule pattern.")
updateToCustomRule(eventRuleName)
println("Updated event rule $eventRuleName to use a custom pattern.")
updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
println("Updated event target $topicArn.")
println(DASHES)

println(DASHES)
println("17. Send an event to trigger the rule. This will trigger a subscription
email.")
triggerCustomRule(email)
println("Events have been sent. Press Enter to continue.")
sc.nextLine()
println(DASHES)

println(DASHES)
println("18. Clean up resources.")
println("Do you want to clean up resources (y/n)")
val ans = sc.nextLine()
if (ans.compareTo("y") == 0) {
    cleanupResources(topicArn, eventRuleName, bucketName, roleName)
```

```
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully completed.")
    println(DASHES)
}

suspend fun cleanupResources(
    topicArn: String?,
    eventRuleName: String?,
    bucketName: String?,
    roleName: String?,
) {
    println("Removing all targets from the event rule.")
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest =
        DetachRolePolicyRequest {
            policyArn = policyArnVal
            roleName = roleNameVal
        }
    IAMClient { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")

        // Delete the role.
        val roleRequest =
            DeleteRoleRequest {
                roleName = roleNameVal
            }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}
```



```
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}
```

```
// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
            }
        }
    }
}
```

```
        eventBrClient.removeTargets(removeTargetsRequest)
        println("Successfully removed the target")
    }
}

suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\", " +
            "\"Message\": \"This event was generated by example code.\" " +
            "\"UtcTime\": \"Now.\" " +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\" "
        }

    val target =
```

```
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern =
        "{" +
            "\"source\": [\"ExampleSource\"]," +
            "\"detail-type\": [\"ExampleType\"]" +
            "}"

    val request =
        PutRuleRequest {
            name = ruleName
            description = "Custom test rule"
            eventPattern = customEventsPattern
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$$.detail.bucket.name"
```

```
myMap["time"] = ".$.time"

val inputTransOb =
    InputTransformer {
        inputTemplate = "\"Notification: an object was uploaded to bucket
<bucket> at <time>.\\""
        inputPathsMap = myMap
    }
val targetOb =
    Target {
        id = targetId
        arn = topicArn
        inputTransformer = inputTransOb
    }

val targetsRequest =
    PutTargetsRequest {
        rule = ruleName
        targets = listOf(targetOb)
        eventBusName = null
    }

EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
    eventBrClient.putTargets(targetsRequest)
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
```

```

        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb =
        PutObjectRequest {
            bucket = bucketName
            key = fileName
            body = myFile.asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {

```

```

    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)
}

```

```
val request =
    PutTargetsRequest {
        eventBusName = null
        targets = targets0b
        rule = ruleName
    }

EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
    eventBrClient.putTargets(request)
    println("Added event rule $eventRuleName with Amazon SNS target $topicName
for bucket $bucketName.")
}
}

suspend fun subEmail(
    topicArnVal: String?,
    email: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Sid\": \"EventBridgePublishTopic\"," +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Resource\": \"*\"," +
```



```

        "\"Action\": \"sns:Publish\" +
        "]" +
    "]"

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
    }
}

suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
// bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """"{

```

```
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }""")

val ruleRequest =
    PutRuleRequest {
        description = "Created by using the AWS SDK for Kotlin"
        name = eventRuleName
        eventPattern = pattern
        roleArn = roleArnVal
    }

EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
    val ruleResponse = eventBrClient.putRule(ruleRequest)
    println("The ARN of the new rule is ${ruleResponse.ruleArn}")
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
    val eventBridgeConfig =
        EventBridgeConfiguration {
        }

    val configuration =
        NotificationConfiguration {
            eventBridgeConfiguration = eventBridgeConfig
        }

    val configurationRequest =
        PutBucketNotificationConfigurationRequest {
            bucket = bucketName
            notificationConfiguration = configuration
            skipDestinationValidation = true
        }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}
```

```
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest =
            HeadBucketRequest {
                bucket = bucketName
            }

        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
}

suspend fun createIAMRole(
    rolenameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
        }
}
```

```
        assumeRolePolicyDocument = polJSON
        description = "Created using the AWS SDK for Kotlin"
    }

    val rolePolicyRequest =
        AttachRolePolicyRequest {
            roleName = rolenameVal
            policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
        }

    IAMClient { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [DeleteRule](#)
 - [DescribeRule](#)
 - [DisableRule](#)
 - [EnableRule](#)
 - [ListRuleNamesByTarget](#)
 - [ListRules](#)
 - [ListTargetsByRule](#)
 - [PutEvents](#)
 - [PutRule](#)
 - [PutTargets](#)

Actions

DeleteRule

L'exemple de code suivant montre comment utiliser `DeleteRule`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteRule](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeRule

L'exemple de code suivant montre comment utiliser `DescribeRule`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }
}
```

```
EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
    val response = eventBrClient.describeRule(ruleRequest)
    println("The state of the rule is $response")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeRule](#) à la section AWS SDK pour la référence de l'API Kotlin.

DisableRule

L'exemple de code suivant montre comment utiliser `DisableRule`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
    }
}
```

```

    }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.enableRule(ruleRequest)
    }
}
}
}

```

- Pour plus de détails sur l'API, reportez-vous [DisableRule](#) à la section AWS SDK pour la référence de l'API Kotlin.

EnableRule

L'exemple de code suivant montre comment utiliser `EnableRule`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
    }
}

```

```

        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [EnableRule](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListRuleNamesByTarget

L'exemple de code suivant montre comment utiliser `ListRuleNamesByTarget`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [ListRuleNamesByTarget](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListRules

L'exemple de code suivant montre comment utiliser `ListRules`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListRules](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListTargetsByRule

L'exemple de code suivant montre comment utiliser `ListTargetsByRule`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTargetsByRule](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutEvents

L'exemple de code suivant montre comment utiliserPutEvents.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun triggerCustomRule(email: String) {
    val json =
```

```

        "{" +
            "\"UserEmail\": \"" + email + "\",\" +
            "\"Message\": \"This event was generated by example code.\" +
            "\"UtcTime\": \"Now.\" +
            "}"

val entry =
    PutEventsRequestEntry {
        source = "ExampleSource"
        detail = json
        detailType = "ExampleType"
    }

val eventsRequest =
    PutEventsRequest {
        this.entries = listOf(entry)
    }

EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
    eventBrClient.putEvents(eventsRequest)
}
}

```

- Pour plus de détails sur l'API, reportez-vous [PutEvents](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutRule

L'exemple de code suivant montre comment utiliser `PutRule`.

SDK pour Kotlin

Note

Il y en a plus 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.

```
suspend fun createScRule(
```

```

    ruleName: String?,
    cronExpression: String?,
) {
    val ruleRequest =
        PutRuleRequest {
            name = ruleName
            eventBusName = "default"
            scheduleExpression = cronExpression
            state = RuleState.Enabled
            description = "A test rule that runs on a schedule created by the Kotlin
API"
        }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

```

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.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"

```

```

        name = eventRuleName
        eventPattern = pattern
        roleArn = roleArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [PutRule](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutTargets

L'exemple de code suivant montre comment utiliser `PutTargets`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }
}

```

```

val targets0b = mutableListOf<Target>()
targets0b.add(myTarget)

val request =
    PutTargetsRequest {
        eventBusName = null
        targets = targets0b
        rule = ruleName
    }

EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
    eventBrClient.putTargets(request)
    println("Added event rule $eventRuleName with Amazon SNS target $topicName
for bucket $bucketName.")
}
}

```

Ajoutez un transformateur d'entrée à une cible pour une règle.

```

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformer0b =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformer0b
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
        }
}

```

```
        eventBusName = null
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutTargets](#) à la section AWS SDK pour la référence de l'API Kotlin.

RemoveTargets

L'exemple de code suivant montre comment utiliser `RemoveTargets`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                    }
            }
        }
    }
}
```

```
        ids = listOf(myTarget.id.toString())
    }
    eventBrClient.removeTargets(removeTargetsRequest)
    println("Successfully removed the target")
}
}
}
```

- Pour plus de détails sur l'API, reportez-vous [RemoveTargets](#) à la section AWS SDK pour la référence de l'API Kotlin.

AWS Glue exemples d'utilisation du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants en utilisant le AWS SDK pour Kotlin avec AWS Glue

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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.

- Répertoriez 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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
            <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
            Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon
            S3) permissions.
            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 (for example,
            cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
            job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
```

```
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("**** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
```

```
        databaseInput = input
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}
```

```
suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }
}
```

```
GlueClient { region = "us-east-1" }.use { glueClient ->
    val response = glueClient.getTables(tableRequest)
    response.tableList?.forEach { tableName ->
        println("Table name is ${tableName.name}")
    }
}

suspend fun startJob(jobNameVal: String?) {
    val runRequest =
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }
}
```

```
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
            maxResults = 10
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
    }
}
```

```
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)

- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

Actions

CreateCrawler

L'exemple de code suivant montre comment utiliser `CreateCrawler`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
```



```
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCrawler](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetCrawler

L'exemple de code suivant montre comment utiliser `GetCrawler`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
}
```

```
GlueClient { region = "us-east-1" }.use { glueClient ->
    val response = glueClient.getCrawler(request)
    val role = response.crawler?.role
    println("The role associated with this crawler is $role")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetCrawler](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetDatabase

L'exemple de code suivant montre comment utiliser `GetDatabase`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetDatabase](#) à la section AWS SDK pour la référence de l'API Kotlin.

StartCrawler

L'exemple de code suivant montre comment utiliser `StartCrawler`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StartCrawler](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples d'IAM utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec IAM.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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 Kotlin

Note

Il y en a plus 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.

```
suspend fun main(args: Array<String>) {
    val usage = """
Usage:
    <username> <policyName> <roleName> <roleSessionName> <fileLocation>
<bucketName>

Where:
    username - The name of the IAM user to create.
    policyName - The name of the policy to create.
    roleName - The name of the role to create.
    roleSessionName - The name of the session required for the assumeRole
operation.
    fileLocation - The file location to the JSON required to create the role
(see Readme).
    bucketName - The name of the Amazon S3 bucket from which objects are read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)
}
```

```
println("*** Getting ready to delete the AWS resources.")
deleteRole(roleName, polArn)
deleteUser(userName)
println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "      \"Effect\": \"Allow\"," +
            "      \"Action\": [" +
            "        \"s3:*\"" +
            "      ]," +
            "      \"Resource\": \"*\"" +
            "    }" +
            "  ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

```
suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IAMClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IAMClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
```

```
        roleName = roleNameVal
        policyArn = policyArnVal
    }
    iamClient.attachRolePolicy(policyRequest)
    println("Successfully attached policy $policyArnVal to role $roleNameVal")
}
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient =
        StsClient {
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken
}
```



```
val staticCredentials =
    StaticCredentialsProvider {
        accessKeyId = key
        secretAccessKey = secKey
        sessionToken = secToken
    }

// List all objects in an Amazon S3 bucket using the temp creds.
val s3 =
    S3Client {
        credentialsProvider = staticCredentials
        region = "us-east-1"
    }

println("Created a S3Client using temp credentials.")
println("Listing objects in $bucketName")

val listObjects =
    ListObjectsRequest {
        bucket = bucketName
    }

val response = s3.listObjects(listObjects)
response.contents?.forEach { myObject ->
    println("The name of the key is ${myObject.key}")
    println("The owner is ${myObject.owner}")
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)
```

```
// Delete the policy.
val request =
    DeletePolicyRequest {
        policyArn = polArn
    }

iam.deletePolicy(request)
println("*** Successfully deleted $polArn")

// Delete the role.
val roleRequest =
    DeleteRoleRequest {
        roleName = roleNameVal
    }

iam.deleteRole(roleRequest)
println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la AWS Référence de l'API de SDK pour Kotlin.
 - [AttachRolePolicy](#)

- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Actions

AttachRolePolicy

L'exemple de code suivant montre comment utiliser `AttachRolePolicy`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IAMClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
    }
}
```

```
    val attachedPolicies = response.attachedPolicies

    // Ensure that the policy is not attached to this role.
    val checkStatus: Int
    if (attachedPolicies != null) {
        checkStatus = checkList(attachedPolicies, policyArnVal)
        if (checkStatus == -1) {
            return
        }
    }

    val policyRequest =
        AttachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }
    iamClient.attachRolePolicy(policyRequest)
    println("Successfully attached policy $policyArnVal to role $roleNameVal")
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- Pour plus de détails sur l'API, reportez-vous [AttachRolePolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateAccessKey

L'exemple de code suivant montre comment utiliser `CreateAccessKey`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateAccessKey](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateAccountAlias

L'exemple de code suivant montre comment utiliser `CreateAccountAlias`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }
}
```

```

    }

    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateAccountAlias](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreatePolicy

L'exemple de code suivant montre comment utiliser `CreatePolicy`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal =
        "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [ " +
            "    { " +
            "      \"Effect\": \"Allow\", " +
            "      \"Action\": [ " +
            "        \"dynamodb:DeleteItem\", " +
            "        \"dynamodb:GetItem\", " +
            "        \"dynamodb:PutItem\", " +
            "        \"dynamodb:Scan\", " +
            "        \"dynamodb:UpdateItem\" " +
            "      ], " +
            "      \"Resource\": \"*\": " +
            "    } " +
            "  ] " +
        "}"
}

```

```
        "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreatePolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateUser

L'exemple de code suivant montre comment utiliser CreateUser.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateUser](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteAccessKey

L'exemple de code suivant montre comment utiliser `DeleteAccessKey`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAccessKey](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteAccountAlias

L'exemple de code suivant montre comment utiliser `DeleteAccountAlias`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteAccountAlias](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeletePolicy

L'exemple de code suivant montre comment utiliser DeletePolicy.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteIAMPolicy(policyARNval: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNval
        }
}
```

```
    }

    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeletePolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteUser

L'exemple de code suivant montre comment utiliser `DeleteUser`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteUser](#) à la section AWS SDK pour la référence de l'API Kotlin.

DetachRolePolicy

L'exemple de code suivant montre comment utiliser `DetachRolePolicy`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetachRolePolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetPolicy

L'exemple de code suivant montre comment utiliser `GetPolicy`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }


    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetPolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListAccessKeys

L'exemple de code suivant montre comment utiliser `ListAccessKeys`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
```

```

        userName = userNameVal
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [ListAccessKeys](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListAccountAliases

L'exemple de code suivant montre comment utiliser `ListAccountAliases`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun listAliases() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [ListAccountAliases](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListUsers

L'exemple de code suivant montre comment utiliser `ListUsers`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listAllUsers() {
    IAMClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListUsers](#) à la section AWS SDK pour la référence de l'API Kotlin.

UpdateUser

L'exemple de code suivant montre comment utiliser `UpdateUser`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateUser](#) à la section AWS SDK pour la référence de l'API Kotlin.

AWS IoT exemples d'utilisation du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants en utilisant le AWS SDK pour Kotlin avec. AWS IoT

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.


Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

 Note

Il y en a plus 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 aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Pour plus de détails sur l'API, voir [ListThings](#) dans le AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)

- [Actions](#)

Principes de base

Principes de base

L'exemple de code suivant montre comment utiliser la gestion des AWS IoT appareils.

SDK pour Kotlin

Note

Il y en a plus 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 aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteString
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
```

```

* Before running this Kotlin code example, ensure that your development environment
* is set up, including configuring your credentials.
*
* For detailed instructions, refer to the following documentation topic:
* [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
*
* This code example requires an SNS topic and an IAM Role.
* Follow the steps in the documentation to set up these resources:
*
* - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-
started.html#step-create-topic)
* - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
*/

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val 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.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")

```

```
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 Kotlin 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 Kotlin environment.
    """).trimIndent(),
)

print("Press Enter to continue...")
scanner.nextLine()
println(DASHES)

println(DASHES)
println("1. Create an AWS IoT thing.")
println(
    """
    An AWS IoT thing represents a virtual entity in the AWS IoT service that can
be associated with a physical device.
    """).trimIndent(),
)
// Prompt the user for input.
print("Enter thing name: ")
thingName = scanner.nextLine()
createIoTThing(thingName)
describeThing(thingName)
println(DASHES)

println(DASHES)
println("2. Generate a device certificate.")
println(
    """
    A device certificate performs a role in securing the communication between
devices (things) and the AWS IoT platform.
    """).trimIndent(),
)
```

```
print("Do you want to create a certificate for $thingName? (y/n)")
val certAns = scanner.nextLine()
var certificateArn: String? = ""
if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
    certificateArn = createCertificate()
    println("Attach the certificate to the AWS IoT thing.")
    attachCertificateToThing(thingName, certificateArn)
} else {
    println("A device certificate was not created.")
}
println(DASHES)

println(DASHES)
println("3. Update an AWS IoT thing with Attributes.")
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.
    """.trimIndent(),
)
print("Press Enter to continue...")
scanner.nextLine()
updateThing(thingName)
println(DASHES)

println(DASHES)
println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
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.
    """.trimIndent(),
)
print("Press Enter to continue...")
scanner.nextLine()
val endpointUrl = describeEndpoint()
println(DASHES)

println(DASHES)
println("5. List your AWS IoT certificates")
```

```

print("Press Enter to continue...")
scanner.nextLine()
if (certificateArn!!.isNotEmpty()) {
    listCertificates()
} else {
    println("You did not create a certificates. Skipping this step.")
}
println(DASHES)

println(DASHES)
println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
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.

        """.trimIndent(),
    )
print("Press Enter to continue...")
scanner.nextLine()
updateShawdowThing(thingName)
println(DASHES)

println(DASHES)
println("7. Write out the state information, in JSON format.")
print("Press Enter to continue...")
scanner.nextLine()
getPayload(thingName)
println(DASHES)

println(DASHES)
println("8. Creates a rule")
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.
        """.trimIndent(),
    )

```

```
print("Enter Rule name: ")
val ruleName = scanner.nextLine()
createIoTRule(roleARN, ruleName, snsAction)
println(DASHES)

println(DASHES)
println("9. List your rules.")
print("Press Enter to continue...")
scanner.nextLine()
listIoTRules()
println(DASHES)

println(DASHES)
println("10. Search things using the name.")
print("Press Enter to continue...")
scanner.nextLine()
val queryString = "thingName:$thingName"
searchThings(queryString)
println(DASHES)

println(DASHES)
if (certificateArn.length > 0) {
    print("Do you want to detach and delete the certificate for $thingName? (y/
n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        println("11. You selected to detach amd delete the certificate.")
        print("Press Enter to continue...")
        scanner.nextLine()
        detachThingPrincipal(thingName, certificateArn)
        deleteCertificate(certificateArn)
    } else {
        println("11. You selected not to delete the certificate.")
    }
} else {
    println("11. You did not create a certificate so there is nothing to
delete.")
}
println(DASHES)

println(DASHES)
println("12. Delete the AWS IoT thing.")
print("Do you want to delete the IoT thing? (y/n)")
```

```
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase = true))
    {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
    { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}
```

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
                ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse = iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
        }
    }
}
```



```

        println("Rule ARN: ${rule.ruleArn}")
        println("-----")
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IoTClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {

```

```
        thingName = thingNameVal
    }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
        iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*)\\.\\.iot\\.\\.us-east-1\\.\\.amazonaws\\.\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
```

```
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
        }
}
```

```
        payload = byteArray
    }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
    }
}
```

```
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

Actions

AttachThingPrincipal

L'exemple de code suivant montre comment utiliser `AttachThingPrincipal`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
```

```
) {  
    val principalRequest =  
        AttachThingPrincipalRequest {  
            thingName = thingNameVal  
            principal = certificateArn  
        }  
  
    IotClient { region = "us-east-1" }.use { iotClient ->  
        iotClient.attachThingPrincipal(principalRequest)  
        println("Certificate attached to $thingNameVal successfully.")  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [AttachThingPrincipal](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateKeysAndCertificate

L'exemple de code suivant montre comment utiliser `CreateKeysAndCertificate`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createCertificate(): String? {  
    IotClient { region = "us-east-1" }.use { iotClient ->  
        val response = iotClient.createKeysAndCertificate()  
        val certificatePem = response.certificatePem  
        val certificateArn = response.certificateArn  
  
        // Print the details.  
        println("\nCertificate:")  
        println(certificatePem)  
        println("\nCertificate ARN:")  
        println(certificateArn)  
        return certificateArn  
    }  
}
```

```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeysAndCertificate](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateThing

L'exemple de code suivant montre comment utiliser `CreateThing`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createIoTThing(thingNameVal: String) {  
    val createThingRequest =  
        CreateThingRequest {  
            thingName = thingNameVal  
        }  
  
    IotClient { region = "us-east-1" }.use { iotClient ->  
        iotClient.createThing(createThingRequest)  
        println("Created $thingNameVal")  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateThing](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateTopicRule

L'exemple de code suivant montre comment utiliser `CreateTopicRule`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IoTClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```


- Pour plus de détails sur l'API, reportez-vous [CreateTopicRule](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteCertificate

L'exemple de code suivant montre comment utiliser `DeleteCertificate`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCertificate](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteThing

L'exemple de code suivant montre comment utiliser `DeleteThing`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteThing](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeEndpoint

L'exemple de code suivant montre comment utiliser `DescribeEndpoint`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
```

```
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeEndpoint](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeThing

L'exemple de code suivant montre comment utiliser `DescribeThing`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeThing](#) à la section AWS SDK pour la référence de l'API Kotlin.

DetachThingPrincipal

L'exemple de code suivant montre comment utiliser `DetachThingPrincipal`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetachThingPrincipal](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListCertificates

L'exemple de code suivant montre comment utiliser `ListCertificates`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListCertificates](#) à la section AWS SDK pour la référence de l'API Kotlin.

SearchIndex

L'exemple de code suivant montre comment utiliser SearchIndex.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }
}
```

```
IotClient { region = "us-east-1" }.use { iotClient ->
    val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
    if (searchIndexResponse.things?.isEmpty() == true) {
        println("No things found.")
    } else {
        searchIndexResponse.things
            ?.forEach { thing -> println("Thing id found using search is
${thing.thingId}") }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SearchIndex](#) à la section AWS SDK pour la référence de l'API Kotlin.

UpdateThing

L'exemple de code suivant montre comment utiliser UpdateThing.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }
}
```

```
val updateThingRequest =
    UpdateThingRequest {
        thingName = thingNameVal
        attributePayload = attributePayloadVal
    }

IotClient { region = "us-east-1" }.use { iotClient ->
    // Update the IoT thing attributes.
    iotClient.updateThing(updateThingRequest)
    println("$thingNameVal attributes updated successfully.")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateThing](#) à la section AWS SDK pour la référence de l'API Kotlin.

AWS IoT data exemples d'utilisation du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants en utilisant le AWS SDK pour Kotlin avec. 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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
            iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetThingShadow](#) à la section AWS SDK pour la référence de l'API Kotlin.

UpdateThingShadow

L'exemple de code suivant montre comment utiliser UpdateThingShadow.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun updateShawdowThing(thingNameVal: String?) {
```



```
// Create the thing shadow state document.
val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity\":50}}}"
val byteStream: ByteStream = ByteStream.fromString(stateDocument)
val byteArray: ByteArray = byteStream.toByteArray()

val updateThingShadowRequest =
    UpdateThingShadowRequest {
        thingName = thingNameVal
        payload = byteArray
    }

IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
    iotPlaneClient.updateThingShadow(updateThingShadowRequest)
    println("The thing shadow was updated successfully.")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateThingShadow](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples d'Amazon Keyspaces utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon Keyspaces.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.


Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

 Note

Il y en a plus 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 Kotlin 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-kotlin/latest/developer-guide/setup.html
*/

suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyspaces](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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 Kotlin

Note

Il y en a plus 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 Kotlin 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-kotlin/latest/developer-guide/setup.html
```

```
This example uses a secure file format to hold certificate information for Kotlin 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 Kotlin 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.

```
*/
```

```
/*
```

```
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.
```

```
*/
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
```

```
        .withConfigLoader(loader)
        .build()

println(DASHES)
println("Welcome to the Amazon Keyspaces example scenario.")
println(DASHES)

println(DASHES)
println("1. Create a keyspace.")
createKeySpace(keyspaceName)
println(DASHES)

println(DASHES)
delay(5000)
println("2. Check for keyspace existence.")
checkKeyspaceExistence(keyspaceName)
println(DASHES)

println(DASHES)
println("3. List keyspaces using a paginator.")
listKeyspacesPaginator()
println(DASHES)

println(DASHES)
println("4. Create a table with a simple movie data schema and enable point-in-
time recovery.")
createTable(keyspaceName, tableName)
println(DASHES)

println(DASHES)
println("5. Check for the table to be in an Active state.")
delay(6000)
checkTable(keyspaceName, tableName)
println(DASHES)

println(DASHES)
println("6. List all tables in the keyspace.")
listTables(keyspaceName)
println(DASHES)

println(DASHES)
println("7. Use a Cassandra driver to insert some records into the Movie
table.")
delay(6000)
```

```
loadData(session, fileName, keyspaceName)
println(DASHES)

println(DASHES)
println("8. Get all records from the Movie table.")
getMovieData(session, keyspaceName)
println(DASHES)

println(DASHES)
println("9. Get a specific Movie.")
getSpecificMovie(session, keyspaceName)
println(DASHES)

println(DASHES)
println("10. Get a UTC timestamp for the current time.")
val utc = ZonedDateTime.now(ZoneOffset.UTC)
println("DATETIME = ${Date.from(utc.toInstant())}")
println(DASHES)

println(DASHES)
println("11. Update the table schema to add a watched Boolean column.")
updateTable(keyspaceName, tableName)
println(DASHES)

println(DASHES)
println("12. Update an item as watched.")
delay(10000) // Wait 10 seconds for the update.
updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
println(DASHES)

println(DASHES)
println("13. Query for items with watched = True.")
getWatchedData(session, keyspaceName)
println(DASHES)

println(DASHES)
println("14. Restore the table back to the previous state using the timestamp.")
println("Note that the restore operation can take up to 20 minutes.")
restoreTable(keyspaceName, utc)
println(DASHES)

println(DASHES)
println("15. Check for completion of the restore action.")
delay(5000)
```

```
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("17. Confirm that both tables are deleted.")
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
```

```
        keyspaceName = keyspaceNameVal
        tableName = tableNameVal
    }

    try {
        KeyspacesClient { region = "us-east-1" }.use { keyClient ->
            // Keep looping until the table cannot be found and a
            ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null
```



```
val tableRequest =
    GetTableRequest {
        keyspaceName = keyspaceNameVal
        tableName = tableNameVal
    }

KeyspacesClient { region = "us-east-1" }.use { keyClient ->
    while (!tableStatus) {
        response = keyClient.getTable(tableRequest)
        status = response!!.status.toString()
        println("The table status is $status")

        if (status.compareTo("ACTIVE") == 0) {
            tableStatus = true
        }
        delay(500)
    }

    val cols = response!!.schemaDefinition?.allColumns
    if (cols != null) {
        for (def in cols) {
            println("The column name is ${def.name}")
            println("The column type is ${def.type}")
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }
}
```

```
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"\$keyspaceName\".\"MovieKotlin\"
WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"\$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0 AND
year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
```

```
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keySpaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keySpaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"\$keySpaceName\".\"MovieKotlin\" WHERE title
= 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is \${item.getString("title")}")
        println("The Movie year is \${item.getInt("year")}")
        println("The plot is \${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keySpaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"\$keySpaceName\".\"MovieKotlin
\";")
    resultSet.forEach { item: Row ->
```

```
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
(:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )
    }
}
```

```
        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
        }
        delay(500)
    }
}
```

```
    }
    val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
    if (cols != null) {
        for (def in cols) {
            println("The column name is ${def.name}")
            println("The column type is ${def.type}")
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
```

```
collList.add(defPlot)

// Set the keys.
val yearKey =
    PartitionKey {
        name = "year"
    }

val titleKey =
    PartitionKey {
        name = "title"
    }

val keyList = ArrayList<PartitionKey>()
keyList.add(yearKey)
keyList.add(titleKey)

val schemaDefinition0b =
    SchemaDefinition {
        partitionKeys = keyList
        allColumns = collList
    }

val timeRecovery =
    PointInTimeRecovery {
        status = PointInTimeRecoveryStatus.Enabled
    }

val tableRequest =
    CreateTableRequest {
        keyspaceName = keySpaceVal
        tableName = tableNameVal
        schemaDefinition = schemaDefinition0b
        pointInTimeRecovery = timeRecovery
    }

KeyspacesClient { region = "us-east-1" }.use { keyClient ->
    val response = keyClient.createTable(tableRequest)
    println("The table ARN is ${response.resourceArn}")
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
```

```
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)

- [GetTable](#)
- [ListKeyspaces](#)
- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

Actions

CreateKeyspace

L'exemple de code suivant montre comment utiliser `CreateKeyspace`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keySpaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keySpaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyspace](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateTable

L'exemple de code suivant montre comment utiliser CreateTable.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
```

```
collList.add(defYear)
collList.add(defReleaseDate)
collList.add(defPlot)

// Set the keys.
val yearKey =
    PartitionKey {
        name = "year"
    }

val titleKey =
    PartitionKey {
        name = "title"
    }

val keyList = ArrayList<PartitionKey>()
keyList.add(yearKey)
keyList.add(titleKey)

val schemaDefinition0b =
    SchemaDefinition {
        partitionKeys = keyList
        allColumns = collList
    }

val timeRecovery =
    PointInTimeRecovery {
        status = PointInTimeRecoveryStatus.Enabled
    }

val tableRequest =
    CreateTableRequest {
        keyspaceName = keySpaceVal
        tableName = tableNameVal
        schemaDefinition = schemaDefinition0b
        pointInTimeRecovery = timeRecovery
    }

KeyspacesClient { region = "us-east-1" }.use { keyClient ->
    val response = keyClient.createTable(tableRequest)
    println("The table ARN is ${response.resourceArn}")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTable](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteKeyspace

L'exemple de code suivant montre comment utiliser `DeleteKeyspace`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyspace](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteTable

L'exemple de code suivant montre comment utiliser `DeleteTable`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTable](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetKeyspace

L'exemple de code suivant montre comment utiliser `GetKeyspace`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
```

```

val keySpaceRequest =
    GetKeyspaceRequest {
        keySpaceName = keySpaceNameVal
    }
KeyspacesClient { region = "us-east-1" }.use { keyClient ->
    val response: GetKeyspaceResponse = keyClient.getKeyspace(keySpaceRequest)
    val name = response.keySpaceName
    println("The $name KeySpace is ready")
}
}

```

- Pour plus de détails sur l'API, reportez-vous [GetKeyspace](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetTable

L'exemple de code suivant montre comment utiliser `GetTable`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun checkTable(
    keySpaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keySpaceName = keySpaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->

```

```

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [GetTable](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListKeyspaces

L'exemple de code suivant montre comment utiliser `ListKeyspaces`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->

```

```
        println("Name: ${obj.keyspaceName}")
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyspaces](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListTables

L'exemple de code suivant montre comment utiliser `ListTables`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTables](#) à la section AWS SDK pour la référence de l'API Kotlin.

RestoreTable

L'exemple de code suivant montre comment utiliser `RestoreTable`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [RestoreTable](#) à la section AWS SDK pour la référence de l'API Kotlin.

UpdateTable

L'exemple de code suivant montre comment utiliser UpdateTable.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateTable](#) à la section AWS SDK pour la référence de l'API Kotlin.

AWS KMS exemples d'utilisation du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants en utilisant le AWS SDK pour Kotlin avec. 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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)

Actions

CreateAlias

L'exemple de code suivant montre comment utiliser `CreateAlias`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
```

```
kmsClient.createAlias(request)
println("$aliasNameVal was successfully created")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateAlias](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateGrant

L'exemple de code suivant montre comment utiliser `CreateGrant`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationObj = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationObj)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateGrant](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateKey

L'exemple de code suivant montre comment utiliser `CreateKey`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKey](#) à la section AWS SDK pour la référence de l'API Kotlin.

Decrypt

L'exemple de code suivant montre comment utiliser `Decrypt`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
```

```
        print(myVal)
    }
}
```

- Pour plus de détails sur l'API, voir [Décrypter](#) dans le AWS SDK pour la référence de l'API Kotlin.

DescribeKey

L'exemple de code suivant montre comment utiliser `DescribeKey`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeKey](#) à la section AWS SDK pour la référence de l'API Kotlin.

DisableKey

L'exemple de code suivant montre comment utiliser `DisableKey`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }


    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DisableKey](#) à la section AWS SDK pour la référence de l'API Kotlin.

EnableKey

L'exemple de code suivant montre comment utiliser `EnableKey`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
```



```
        keyId = keyIdVal
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [EnableKey](#) à la section AWS SDK pour la référence de l'API Kotlin.

Encrypt

L'exemple de code suivant montre comment utiliser `Encrypt`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
    }
}
```

```
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Pour plus de détails sur l'API, voir [Encrypt](#) in AWS SDK for Kotlin API reference.

ListAliases

L'exemple de code suivant montre comment utiliser `ListAliases`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }
}
```

```
KmsClient { region = "us-west-2" }.use { kmsClient ->
    val response = kmsClient.listAliases(request)
    response.aliases?.forEach { alias ->
        println("The alias name is ${alias.aliasName}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListAliases](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListGrants

L'exemple de code suivant montre comment utiliser `ListGrants`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListGrants](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListKeys

L'exemple de code suivant montre comment utiliser `ListKeys`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeys](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples Lambda utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Lambda.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Principes de base](#)
- [Actions](#)
- [Scénarios](#)

Principes de base

Principes de base

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 Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>

        Where:
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that
            has AWS Lambda permissions.
            handler - The fully qualified method name (for example,
            example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
            that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
            or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file (for
            example, LambdaHello-1.0-SNAPSHOT.jar).
        """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val functionName = args[0]
    val role = args[1]
    val handler = args[2]
    val bucketName = args[3]
    val updatedBucketName = args[4]
    val key = args[5]

    println("Creating a Lambda function named $functionName.")
    val funArn = createScFunction(functionName, bucketName, key, handler, role)
    println("The AWS Lambda ARN is $funArn")
}
```

```
// Get a specific Lambda function.
println("Getting the $functionName AWS Lambda function.")
getFunction(functionName)

// List the Lambda functions.
println("Listing all AWS Lambda functions.")
listFunctionsSc()

// Invoke the Lambda function.
println("*** Invoke the Lambda function.")
invokeFunctionSc(functionName)

// Update the AWS Lambda function code.
println("*** Update the Lambda function code.")
updateFunctionCode(functionName, updatedBucketName, key)

// println("*** Invoke the function again after updating the code.")
invokeFunctionSc(functionName)

// Update the AWS Lambda function configuration.
println("Update the run time of the function.")
updateFunctionConfiguration(functionName, handler)

// Delete the AWS Lambda function.
println("Delete the AWS Lambda function.")
delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
```

```
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java17
    }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
}

suspend fun getFunction(functionNameVal: String) {
    val functionRequest =
        GetFunctionRequest {
            functionName = functionNameVal
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
        ${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {
    val request =
        ListFunctionsRequest {
            maxItems = 10
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}
```



```
suspend fun invokeFunctionSc(functionNameVal: String) {
    val json = """"{"inputValue":"1000}""""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            payload = byteArray
            logType = LogType.Tail
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(
    functionNameVal: String?,
    bucketName: String?,
    key: String?,
) {
    val functionCodeRequest =
        UpdateFunctionCodeRequest {
            functionName = functionNameVal
            publish = true
            s3Bucket = bucketName
            s3Key = key
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun updateFunctionConfiguration(
    functionNameVal: String?,
    handlerVal: String?,
) {
    val configurationRequest =
```

```
        UpdateFunctionConfigurationRequest {
            functionName = functionNameVal
            handler = handlerVal
            runtime = Runtime.Java17
        }

        LambdaClient { region = "us-east-1" }.use { awsLambda ->
            awsLambda.updateFunctionConfiguration(configurationRequest)
        }
    }

suspend fun delFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

Actions

CreateFunction

L'exemple de code suivant montre comment utiliser `CreateFunction`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String? {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateFunction](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteFunction

L'exemple de code suivant montre comment utiliser `DeleteFunction`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteFunction](#) à la section AWS SDK pour la référence de l'API Kotlin.

Invoke

L'exemple de code suivant montre comment utiliser `Invoke`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun invokeFunction(functionNameVal: String) {
    val json = """"{"inputValue":"1000}""""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            logType = LogType.Tail
            payload = byteArray
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
        println("The log result is ${res.logResult}")
    }
}
```

- Pour plus d'informations sur l'API, consultez [Invoke](#) dans la AWS Référence d'API du kit SDK pour Kotlin.

Scénarios

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

SDK pour Kotlin

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

MediaConvert exemples d'utilisation du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants en utilisant le AWS SDK pour Kotlin avec 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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInputVal: String,
): String? {
    val s3path = fileInputVal.substring(0, fileInputVal.lastIndexOf('/') + 1) +
    "javasdk/out/"
    val fileOutput = s3path + "index"
    val thumbsOutput = s3path + "thumbs/"
    val mp4Output = s3path + "mp4/"

    try {
        val describeEndpoints =
            DescribeEndpointsRequest {
                maxResults = 20
            }

        val res = mcClient.describeEndpoints(describeEndpoints)
        if (res.endpoints?.size!! <= 0) {
            println("Cannot find MediaConvert service endpoint URL!")
            exitProcess(0)
        }
        val endpointURL = res.endpoints!!.get(0).url!!
        val mediaConvert =
            MediaConvertClient.fromEnvironment {
                region = "us-west-2"
                endpointProvider =
                    MediaConvertEndpointProvider {
                        Endpoint(endpointURL)
                    }
            }

        // output group Preset HLS low profile
        val hlsLow = createOutput("_low", "_\${dt}", 750000, 7, 1920, 1080, 640)
```

```

// output group Preset HLS medium profile
val hlsMedium = createOutput("_medium", "_\${dt$}", 1200000, 7, 1920, 1080,
1280)

// output group Preset HLS high profole
val hlsHigh = createOutput("_high", "_\${dt$}", 3500000, 8, 1920, 1080, 1920)

val outputSettings =
    OutputGroupSettings {
        type = OutputGroupType.HlsGroupSettings
    }

val outputObsList: MutableList<Output> = mutableListOf()
if (hlsLow != null) {
    outputObsList.add(hlsLow)
}
if (hlsMedium != null) {
    outputObsList.add(hlsMedium)
}
if (hlsHigh != null) {
    outputObsList.add(hlsHigh)
}

// Create an OutputGroup object.
val appleHLS =
    OutputGroup {
        name = "Apple HLS"
        customName = "Example"
        outputGroupSettings =
            OutputGroupSettings {
                type = OutputGroupType.HlsGroupSettings
                this.hlsGroupSettings =
                    HlsGroupSettings {
                        directoryStructure =
HlsDirectoryStructure.SingleDirectory
                        manifestDurationFormat =
HlsManifestDurationFormat.Integer
                        streamInfResolution = HlsStreamInfResolution.Include
                        clientCache = HlsClientCache.Enabled
                        captionLanguageSetting =
HlsCaptionLanguageSetting.Omit
                        manifestCompression = HlsManifestCompression.None
                        codecSpecification = HlsCodecSpecification.Rfc4281

```



```

        outputSelection =
HlsOutputSelection.ManifestsAndSegments
        programDateTime = HlsProgramDateTime.Exclude
        programDateTimePeriod = 600
        timedMetadataId3Frame =
HlsTimedMetadataId3Frame.Priv
        timedMetadataId3Period = 10
        destination = fileOutput
        segmentControl = HlsSegmentControl.SegmentedFiles
        minFinalSegmentLength = 0.toDouble()
        segmentLength = 4
        minSegmentLength = 1
    }
}
outputs = outputObsList
}

val theOutput =
Output {
    extension = "mp4"
    containerSettings =
        ContainerSettings {
            container = ContainerType.fromValue("MP4")
        }
}

videoDescription =
VideoDescription {
    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 {
            codec = VideoCodec.H264
            h264Settings =
                H264Settings {
                    rateControlMode = H264RateControlMode.Qvbr

```

```

        parControl =
H264ParControl.InitializeFromSource
        qualityTuningLevel =
H264QualityTuningLevel.SinglePass
        qvbrSettings = H264QvbrSettings
    { qvbrQualityLevel = 8 }
        codecLevel = H264CodecLevel.Auto
        codecProfile = H264CodecProfile.Main
        maxBitrate = 2400000
        framerateControl =
H264FramerateControl.InitializeFromSource
        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.DuplicateDrop
        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
    }

```

```

        }
    }

    audioDescriptions =
        listOf(
            AudioDescription {
                audioTypeControl = AudioTypeControl.FollowInput
                languageCodeControl =
AudioLanguageCodeControl.FollowInput
                codecSettings =
                    AudioCodecSettings {
                        codec = AudioCodec.Aac
                        aacSettings =
                            AacSettings {
                                codecProfile = AacCodecProfile.Lc
                                rateControlMode = AacRateControlMode.Cbr
                                codingMode = AacCodingMode.CodingMode2_0
                                sampleRate = 44100
                                bitrate = 160000
                                rawFormat = AacRawFormat.None
                                specification = AacSpecification.Mpeg4
                                audioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.Normal
                            }
                        }
                    },
        )
    }

    // Create an OutputGroup
    val fileMp4 =
        OutputGroup {
            name = "File Group"
            customName = "mp4"
            outputGroupSettings =
                OutputGroupSettings {
                    type = OutputGroupType.FileGroupSettings
                    fileGroupSettings =
                        FileGroupSettings {
                            destination = mp4Output
                        }
                }
            outputs = listOf(theOutput)
        }
    }

```

```
val containerSettings1 =
    ContainerSettings {
        container = ContainerType.Raw
    }

val thumbs =
    OutputGroup {
        name = "File Group"
        customName = "thumbs"
        outputGroupSettings =
            OutputGroupSettings {
                type = OutputGroupType.FileGroupSettings
                fileGroupSettings =
                    FileGroupSettings {
                        destination = thumbsOutput
                    }
            }

        outputs =
            listOf(
                Output {
                    extension = "jpg"

                    this.containerSettings = containerSettings1
                    videoDescription =
                        VideoDescription {
                            scalingBehavior = ScalingBehavior.Default
                            sharpness = 50
                            antiAlias = AntiAlias.Enabled
                            timecodeInsertion =
                                VideoTimecodeInsertion.Disabled

                            colorMetadata = ColorMetadata.Insert
                            dropFrameTimecode = DropFrameTimecode.Enabled
                            codecSettings =
                                VideoCodecSettings {
                                    codec = VideoCodec.FrameCapture
                                    frameCaptureSettings =
                                        FrameCaptureSettings {
                                            framerateNumerator = 1
                                            framerateDenominator = 1
                                            maxCaptures = 10000000
                                            quality = 80
                                        }
                                }
                }
            )
    }
```

```

        }
    },
)
}

val audioSelectors1: MutableMap<String, AudioSelector> = HashMap()
audioSelectors1["Audio Selector 1"] =
    AudioSelector {
        defaultSelection = AudioDefaultSelection.Default
        offset = 0
    }

val jobSettings =
    JobSettings {
        inputs =
            listOf(
                Input {
                    audioSelectors = audioSelectors1
                    videoSelector =
                        VideoSelector {
                            colorSpace = ColorSpace.Follow
                            rotate = InputRotate.Degree0
                        }
                    filterEnable = InputFilterEnable.Auto
                    filterStrength = 0
                    deblockFilter = InputDeblockFilter.Disabled
                    denoiseFilter = InputDenoiseFilter.Disabled
                    psiControl = InputPsiControl.UsePsi
                    timecodeSource = InputTimecodeSource.Embedded
                    fileInput = fileInputVal

                    outputGroups = listOf(appleHLS, thumbs, fileMp4)
                },
            )
    }

val createJobRequest =
    CreateJobRequest {
        role = mcRoleARN
        settings = jobSettings
    }

val createJobResponse = mediaConvert.createJob(createJobRequest)

```

```
        return createJobResponse.job?.id
    } catch (ex: MediaConvertException) {
        println(ex.message)
        mcClient.close()
        exitProcess(0)
    }
}

fun createOutput(
    nameModifierVal: String,
    segmentModifierVal: String,
    qvbrMaxBitrate: Int,
    qvbrQualityLevelVal: Int,
    originWidth: Int,
    originHeight: Int,
    targetWidth: Int,
): Output? {
    val targetHeight = (
        (originHeight * targetWidth / originWidth).toFloat().roundToInt() -
        (originHeight * targetWidth / originWidth).toFloat().roundToInt() % 4
    )

    var output: Output?
    try {
        val audio1 =
            AudioDescription {
                audioTypeControl = AudioTypeControl.FollowInput
                languageCodeControl = AudioLanguageCodeControl.FollowInput
                codecSettings =
                    AudioCodecSettings {
                        codec = AudioCodec.Aac
                        aacSettings =
                            AacSettings {
                                codecProfile = AacCodecProfile.Lc
                                rateControlMode = AacRateControlMode.Cbr
                                codingMode = AacCodingMode.CodingMode2_0
                                sampleRate = 44100
                                bitrate = 96000
                                rawFormat = AacRawFormat.None
                                specification = AacSpecification.Mpeg4
                                audioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.Normal
                            }
                    }
            }
    }
}
```

```
    }

    output =
        Output {
            nameModifier = nameModifierVal
            outputSettings =
                OutputSettings {
                    hlsSettings =
                        HlsSettings {
                            segmentModifier = segmentModifierVal
                            audioGroupId = "program_audio"
                            iFrameOnlyManifest = HlsIFrameOnlyManifest.Exclude
                        }
                }
            containerSettings =
                ContainerSettings {
                    container = ContainerType.M3U8
                    this.m3u8Settings =
                        M3u8Settings {
                            audioFramesPerPes = 4
                            pcrControl = M3u8PcrControl.PcrEveryPesPacket
                            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 = listOf(482, 483, 484, 485, 486, 487,
488, 489, 490, 491, 492)
                        }

                    videoDescription =
                        VideoDescription {
                            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 {
        codec = VideoCodec.H264
        h264Settings =
            H264Settings {
                rateControlMode =
H264RateControlMode.Qvbr
                parControl =
H264ParControl.InitializeFromSource
                qualityTuningLevel =
H264QualityTuningLevel.SinglePass
                qvbrSettings =
                    H264QvbrSettings {
                        qvbrQualityLevel =
qvbrQualityLevelVal
                    }
                codecLevel = H264CodecLevel.Auto
                codecProfile =
                    if (targetHeight > 720 &&
                        targetWidth > 1280
                    ) {
                        H264CodecProfile.High
                    } else {
                        H264CodecProfile.Main
                    }
                maxBitrate = qvbrMaxBitrate
                framerateControl =
H264FramerateControl.InitializeFromSource
                gopSize = 2.0
                gopSizeUnits =
H264GopSizeUnits.Seconds
                numberBframesBetweenReferenceFrames
= 2
                gopClosedCadence = 1
                gopBReference =
H264GopBReference.Disabled
                slowPal = H264SlowPal.Disabled
                syntax = H264Syntax.Default
                numberReferenceFrames = 3

```



```

        H264DynamicSubGop.Static
        H264FieldEncoding.Paff
        H264SceneChangeDetect.Enabled
        H264FramerateConversionAlgorithm.DuplicateDrop
        H264EntropyEncoding.Cabac
        H264UnregisteredSeiTimecode.Disabled
        H264AdaptiveQuantization.High
        H264SpatialAdaptiveQuantization.Enabled
        H264TemporalAdaptiveQuantization.Enabled
        H264FlickerAdaptiveQuantization.Disabled
        H264InterlaceMode.Progressive
    }
    }
    audioDescriptions = listOf(audio1)
}
} catch (ex: MediaConvertException) {
    println(ex.toString())
    exitProcess(0)
}
return output
}
dynamicSubGop =
fieldEncoding =
sceneChangeDetect =
minIInterval = 0
telecine = H264Telecine.None
framerateConversionAlgorithm =
entropyEncoding =
slices = 1
unregisteredSeiTimecode =
repeatPps = H264RepeatPps.Disabled
adaptiveQuantization =
spatialAdaptiveQuantization =
temporalAdaptiveQuantization =
flickerAdaptiveQuantization =
softness = 0
interlaceMode =

```

- Pour plus de détails sur l'API, reportez-vous [CreateJob](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetJob

L'exemple de code suivant montre comment utiliser `GetJob`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getSpecificJob(
    mcClient: MediaConvertClient,
    jobId: String?,
) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }

    val endpointURL = res.endpoints!!.get(0).url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobRequest =
        GetJobRequest {
            id = jobId
        }

    val response: GetJobResponse = mediaConvert.getJob(jobRequest)
```

```
println("The ARN of the job is ${response.job?.arn}.")
}
```

- Pour plus de détails sur l'API, reportez-vous [GetJob](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListJobs

L'exemple de code suivant montre comment utiliser `ListJobs`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listCompleteJobs(mcClient: MediaConvertClient) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
    val endpointURL = res.endpoints!![0].url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobsRequest =
```

```
ListJobsRequest {
    maxResults = 10
    status = JobStatus.fromValue("COMPLETE")
}

val jobsResponse = mediaConvert.listJobs(jobsRequest)
val jobs = jobsResponse.jobs
if (jobs != null) {
    for (job in jobs) {
        println("The JOB ARN is ${job.arn}")
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListJobs](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples d'Amazon Pinpoint utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec 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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequest0b =
        CreateApplicationRequest {
            name = applicationName
        }


    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequest0b
                },
            )
        return result.applicationResponse?.id
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateApp](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateCampaign

L'exemple de code suivant montre comment utiliser `CreateCampaign`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val schedule0b =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessage0b =
        Message {
            action = Action.OpenApp
            body = "My message body"
            title = "My message title"
        }

    val messageConfiguration0b =
        MessageConfiguration {
            defaultMessage = defaultMessage0b
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = schedule0b
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfiguration0b
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCampaign](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateSegment

L'exemple de code suivant montre comment utiliser `CreateSegment`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {
    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts =
        AttributeDimension {
            attributeType = AttributeType.Inclusive
            values = myList
        }

    segmentAttributes["Team"] = atts
    val recencyDimension =
        RecencyDimension {
            duration = Duration.fromValue("DAY_30")
            recencyType = RecencyType.fromValue("ACTIVE")
        }

    val segmentBehaviors =
        SegmentBehaviors {
            recency = recencyDimension
        }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb =
        SegmentDimensions {
            attributes = segmentAttributes
        }
}
```

```

        behavior = segmentBehaviors
        demographic = SegmentDemographics {}
        location = segmentLocation
    }

    val writeSegmentRequest0b =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensions0b
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequest0b
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateSegment](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteApp

L'exemple de code suivant montre comment utiliser `DeleteApp`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deletePinApp(appId: String?) {
```



```

PinpointClient { region = "us-west-2" }.use { pinpoint ->
    val result =
        pinpoint.deleteApp(
            DeleteAppRequest {
                applicationId = appId
            },
        )
    val appName = result.applicationResponse?.name
    println("Application $appName has been deleted.")
}
}

```

- Pour plus de détails sur l'API, reportez-vous [DeleteApp](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteEndpoint

L'exemple de code suivant montre comment utiliser DeleteEndpoint.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
    }
}

```

```
        println("The deleted endpoint is $id")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteEndpoint](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetEndpoint

L'exemple de code suivant montre comment utiliser `GetEndpoint`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.getEndpoint(
                GetEndpointRequest {
                    applicationId = appId
                    endpointId = endpoint
                },
            )
        val endResponse = result.endpointResponse

        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson =
            GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create()
    }
}
```

```
        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetEndpoint](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetSegments

L'exemple de code suivant montre comment utiliser `GetSegments`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listSegs(appId: String?) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segment id is ${segment.id}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetSegments](#) à la section AWS SDK pour la référence de l'API Kotlin.

SendMessage

L'exemple de code suivant montre comment utiliser `SendMessage`.

SDK pour Kotlin

Note

Il y en a plus 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 Kotlin 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-kotlin/latest/developer-guide/setup.html
*/

val body: String =
    """
    Amazon Pinpoint test (AWS SDK for Kotlin)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK for
    Kotlin.

    """.trimIndent()

suspend fun main(args: Array<String>) {
    val 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
        toAddress - The to address. This address has to be verified in Amazon
    Pinpoint in the region you're using to send email
    """
}
```

```
    if (args.size != 3) {
        println(usage)
        exitProcess(0)
    }

    val subject = args[0]
    val senderAddress = args[1]
    val toAddress = args[2]
    sendEmail(subject, senderAddress, toAddress)
}

suspend fun sendEmail(
    subjectVal: String?,
    senderAddress: String,
    toAddressVal: String,
) {
    var content =
        Content {
            data = body
        }

    val messageBody =
        Body {
            text = content
        }

    val subContent =
        Content {
            data = subjectVal
        }

    val message =
        Message {
            body = messageBody
            subject = subContent
        }

    val destination0b =
        Destination {
            toAddresses = listOf(toAddressVal)
        }

    val emailContent =
        EmailContent {
```

```
        simple = message
    }

    val sendEmailRequest =
        SendEmailRequest {
            fromEmailAddress = senderAddress
            destination = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient { region = "us-east-1" }.use { pinpointemail ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SendMessages](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples d'Amazon RDS utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon RDS.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Principes de base](#)

- [Actions](#)
- [Scénarios](#)

Principes de base

Principes de base

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 Kotlin

Note

Il y en a plus 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 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-kotlin/latest/developer-guide/setup.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
```

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking the createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment parameters by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance that contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.

```
*/
```

```
var sleepTime: Long = 20
```

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>
            <dbSnapshotIdentifier><secretName>
```

```
    Where:
```

```
        dbGroupName - The database group name.
```

```
        dbParameterGroupFamily - The database parameter group name.
```

```
        dbInstanceIdentifier - The database instance identifier.
```

```
        dbName - The database name.
```

```
        dbSnapshotIdentifier - The snapshot identifier.
```

```
        secretName - The name of the AWS Secrets Manager secret that contains
```

```
the database credentials.
```

```
    """
```

```
    if (args.size != 6) {
        println(usage)
```



```
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val dbSnapshotIdentifier = args[4]
    val secretName = args[5]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeDBEngines()

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)

    println("8. Get a list of micro instance classes available for the selected
engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySQL database and
uses the parameter group")
```

```
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                }
            }
        }
    }
}
```

```

        if (instanceARN.compareTo(dbARN) == 0) {
            println("$dbARN still exists")
            didFind = true
        }
        if (index == listSize && !didFind) {
            // Went through the entire list and did not find the
database name.
                isDataDel = true
            }
            index++
        }
    }
}

// Delete the para group.
val parameterGroupRequest =
    DeleteDbParameterGroupRequest {
        dbParameterGroupName = dbGroupName
    }
rdsClient.deleteDbParameterGroup(parameterGroupRequest)
println("$dbGroupName was deleted.")
}
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
    ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {

```

```
var snapshotReady = false
var snapshotReadyStr: String
println("Waiting for the snapshot to become available.")

val snapshotsRequest =
    DescribeDbSnapshotsRequest {
        dbSnapshotIdentifier = dbSnapshotIdentifierVal
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }

while (!snapshotReady) {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbSnapshots(snapshotsRequest)
        val snapshotList: List<DbSnapshot>? = response.dbSnapshots
        if (snapshotList != null) {
            for (snapshot in snapshotList) {
                snapshotReadyStr = snapshot.status.toString()
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
}
println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}
```

```

    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =

```

```
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.0.35"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.createDbInstance(instanceRequest)
            print("The status is ${response.dbInstance?.dbInstanceStatus}")
            return response.dbInstance?.dbInstanceArn
        }
    }

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
            rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
        }
}
```

```
        engine = "mysql"
    }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
successfully modified")
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
```

```

    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paramName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset or
                auto_increment_increment.
                paramName = para.parameterName.toString()
                if (paramName.compareTo("auto_increment_offset") == 0 ||
                paramName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is $paramName")
                    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}")
                }
            }
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->

```



```

    val response = rdsClient.describeDbParameterGroups(groupsRequest)
    val groups = response.dbParameterGroups
    if (groups != null) {
        for (group in groups) {
            println("The group name is ${group.dbParameterGroupName}")
            println("The group description is ${group.description}")
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
    ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.

```

```
        if (engines != null) {
            for (engine0b in engines) {
                println("The name of the DB parameter group family for the database
engine is ${engine0b.dbParameterGroupFamily}.")
                println("The name of the database engine ${engine0b.engine}.")
                println("The version number of the database engine
${engine0b.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-west-2" }.use { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CréerDBInstance](#)
 - [Créer un DBParameter groupe](#)
 - [CréerDBSnapshot](#)
 - [SuppressionDBInstance](#)
 - [Supprimer le DBParameter groupe](#)
 - [Décrire DBEngine les versions](#)
 - [Décrivez DBInstances](#)
 - [Décrire DBParameter les groupes](#)
 - [Décrivez DBParameters](#)
 - [Décrivez DBSnapshots](#)
 - [DescribeOrderableDBInstanceOptions](#)

- [Modifier le DBParameter groupe](#)

Actions

CreateDBInstance

L'exemple de code suivant montre comment utiliser CreateDBInstance.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro" // Use a supported instance class
            engineVersion = "8.0.39" // Use a supported engine version
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

```
// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- Pour plus de détails sur l'API, voir [Create DBInstance](#) in AWS SDK for Kotlin API reference.

DeleteDBInstance

L'exemple de code suivant montre comment utiliser `DeleteDBInstance`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
        ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Pour plus de détails sur l'API, voir [Supprimer DBInstance dans le AWS SDK](#) pour la référence de l'API Kotlin.

DescribeAccountAttributes

L'exemple de code suivant montre comment utiliser DescribeAccountAttributes.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getAccountAttributes() {
```

```
RdsClient { region = "us-west-2" }.use { rdsClient ->
    val response =
    rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAccountAttributes](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeDBInstances

L'exemple de code suivant montre comment utiliser `DescribeDBInstances`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeInstances() {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest {})
        response.dbInstances?.forEach { instance ->
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- Pour plus de détails sur l'API, voir [Description DBInstances](#) dans le AWS SDK pour la référence de l'API Kotlin.

ModifyDBInstance

L'exemple de code suivant montre comment utiliser `ModifyDBInstance`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
        ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- Pour plus de détails sur l'API, voir [Modifier DBInstance](#) dans le AWS SDK pour la référence de l'API Kotlin.

Scénarios

Créer un outil de suivi des éléments de travail sans serveur Aurora

L'exemple de code suivant montre comment créer une application Web qui suit les éléments de travail dans une base de données Amazon Aurora Serverless et utilise Amazon Simple Email Service (Amazon SES) pour envoyer des rapports.

SDK pour Kotlin

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](#).

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Exemples d'Amazon RDS Data Service utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon RDS Data Service.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Scénarios](#)

Scénarios

Créer un outil de suivi des éléments de travail sans serveur Aurora

L'exemple de code suivant montre comment créer une application Web qui suit les éléments de travail dans une base de données Amazon Aurora Serverless et utilise Amazon Simple Email Service (Amazon SES) pour envoyer des rapports.

SDK pour Kotlin

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](#).

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Exemples d'Amazon Redshift utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec 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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateCluster

L'exemple de code suivant montre comment utiliser `CreateCluster`.

SDK pour Kotlin

Note

Il y en a plus 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 .

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCluster](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteCluster

L'exemple de code suivant montre comment utiliser `DeleteCluster`.

SDK pour Kotlin

Note

Il y en a plus 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.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCluster](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeClusters

L'exemple de code suivant montre comment utiliser `DescribeClusters`.

SDK pour Kotlin

 Note

Il y en a plus 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.

```
suspend fun describeRedshiftClusters() {
    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse =
            redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters


        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeClusters](#) à la section AWS SDK pour la référence de l'API Kotlin.

ModifyCluster

L'exemple de code suivant montre comment utiliser `ModifyCluster`.

SDK pour Kotlin

 Note

Il y en a plus 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.

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
            "The modified cluster was successfully modified and has
            ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance window",
        )
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ModifyCluster](#) à la section AWS SDK pour la référence de l'API Kotlin.

Scénarios

Créer une application web pour suivre les données Amazon Redshift

L'exemple de code suivant montre comment créer une application Web qui suit et génère des rapports sur les éléments de travail à l'aide d'une base de données Amazon Redshift.

SDK pour Kotlin

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

Exemples d'Amazon Rekognition utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec 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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun compareTwoFaces(  
    similarityThresholdVal: Float,  
    sourceImageVal: String,  
    targetImageVal: String,
```

```
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches with
                    ${face.confidence} % confidence.")
                }
            }
        }

        val uncomparated = compareFacesResult.unmatchedFaces
        if (uncomparated != null) {
            println("There was ${uncomparated.size} face(s) that did not match")
        }
    }
}
```

```
println("Source image rotation:
${compareFacesResult.sourceImageOrientationCorrection}")
println("target image rotation:
${compareFacesResult.targetImageOrientationCorrection}")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CompareFaces](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateCollection

L'exemple de code suivant montre comment utiliser `CreateCollection`.

Pour plus d'informations, consultez [Création d'une collection](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCollection](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteCollection

L'exemple de code suivant montre comment utiliser `DeleteCollection`.

Pour plus d'informations, consultez [Suppression d'une collection](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteCollection](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteFaces

L'exemple de code suivant montre comment utiliser `DeleteFaces`.

Pour plus d'informations, veuillez consulter [Supprimer des visages d'une collection](#).

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteFaces](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeCollection

L'exemple de code suivant montre comment utiliser `DescribeCollection`.

Pour plus d'informations, veuillez consulter [Description d'une collection](#).

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCollection](#) à la section AWS SDK pour la référence de l'API Kotlin.

DetectFaces

L'exemple de code suivant montre comment utiliser DetectFaces.

Pour plus d'informations, veuillez consulter [Détection des visages dans une image](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }
}
```

```
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between ${ageRange?.low}
and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectFaces](#) à la section AWS SDK pour la référence de l'API Kotlin.

DetectLabels

L'exemple de code suivant montre comment utiliser `DetectLabels`.

Pour plus d'informations, veuillez consulter [Détection des étiquettes dans une image](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }
}
```

```
RekognitionClient { region = "us-east-1" }.use { rekClient ->
    val response = rekClient.detectLabels(request)
    response.labels?.forEach { label ->
        println("${label.name} : ${label.confidence}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectLabels](#) à la section AWS SDK pour la référence de l'API Kotlin.

DetectModerationLabels

L'exemple de code suivant montre comment utiliser `DetectModerationLabels`.

Pour plus d'informations, veuillez consulter [Détection des images inappropriées](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
    }
}
```

```

        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
Parent: ${label.parentName}")
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [DetectModerationLabels](#) à la section AWS SDK pour la référence de l'API Kotlin.

DetectText

L'exemple de code suivant montre comment utiliser `DetectText`.

Pour plus d'informations, consultez [Détection de texte dans une image](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
        }
    }
}

```

```
        println("Parent Id: ${text.parentId}")
        println("Type: ${text.type}")
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DetectText](#) à la section AWS SDK pour la référence de l'API Kotlin.

IndexFaces

L'exemple de code suivant montre comment utiliser `IndexFaces`.

Pour plus d'informations, veuillez consulter [Ajouter des visages à une collection](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }
}
```

```

RekognitionClient { region = "us-east-1" }.use { rekClient ->
    val facesResponse = rekClient.indexFaces(request)

    // Display the results.
    println("Results for the image")
    println("\n Faces indexed:")
    facesResponse.faceRecords?.forEach { faceRecord ->
        println("Face ID: ${faceRecord.face?.faceId}")
        println("Location: ${faceRecord.faceDetail?.boundingBox}")
    }

    println("Faces not indexed:")
    facesResponse.unindexedFaces?.forEach { unindexedFace ->
        println("Location: ${unindexedFace.faceDetail?.boundingBox}")
        println("Reasons:")

        unindexedFace.reasons?.forEach { reason ->
            println("Reason: $reason")
        }
    }
}
}
}

```

- Pour plus de détails sur l'API, reportez-vous [IndexFaces](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListCollections

L'exemple de code suivant montre comment utiliser `ListCollections`.

Pour en savoir plus, consultez [Répertoire de collections](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listAllCollections() {
```



```
val request =
    ListCollectionsRequest {
        maxResults = 10
    }

RekognitionClient { region = "us-east-1" }.use { rekClient ->
    val response = rekClient.listCollections(request)
    response.collectionIds?.forEach { resultId ->
        println(resultId)
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListCollections](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListFaces

L'exemple de code suivant montre comment utiliser `ListFaces`.

Pour plus d'informations, consultez [Répertoire de visages d'une collection](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
        }
    }
}
```

```
        println("The face Id value is ${face.faceId}")
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListFaces](#) à la section AWS SDK pour la référence de l'API Kotlin.

RecognizeCelebrities

L'exemple de code suivant montre comment utiliser `RecognizeCelebrities`.

Pour plus d'informations, consultez [Reconnaissance de célébrités dans une image](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
    }
}
```

```
        }  
    }  
    println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")  
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [RecognizeCelebrities](#) à la section AWS SDK pour la référence de l'API Kotlin.

Scénarios

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

SDK pour Kotlin

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

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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Détectez des visages dans une vidéo stockée dans un compartiment Amazon S3.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidObj =
        Video {
            s3Object = s3obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidObj
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}
```

```
    }  
}  
  
suspend fun getFaceResults() {  
    var finished = false  
    var status: String  
    var yy = 0  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        var response: GetFaceDetectionResponse? = null  
  
        val recognitionRequest =  
            GetFaceDetectionRequest {  
                jobId = startJobId  
                maxResults = 10  
            }  
  
        // Wait until the job succeeds.  
        while (!finished) {  
            response = rekClient.getFaceDetection(recognitionRequest)  
            status = response.jobStatus.toString()  
            if (status.compareTo("Succeeded") == 0) {  
                finished = true  
            } else {  
                println("$yy status is: $status")  
                delay(1000)  
            }  
            yy++  
        }  
  
        // Proceed when the job is done - otherwise VideoMetadata is null.  
        val videoMetaData = response?.videoMetadata  
        println("Format: ${videoMetaData?.format}")  
        println("Codec: ${videoMetaData?.codec}")  
        println("Duration: ${videoMetaData?.durationMillis}")  
        println("FrameRate: ${videoMetaData?.frameRate}")  
  
        // Show face information.  
        response?.faces?.forEach { face ->  
            println("Age: ${face.face?.ageRange}")  
            println("Face: ${face.face?.beard}")  
            println("Eye glasses: ${face?.face?.eyeglasses}")  
            println("Mustache: ${face.face?.mustache}")  
            println("Smile: ${face.face?.smile}")  
        }  
    }  
}
```

```
}  
}
```

Détectez un contenu inapproprié ou offensant dans une vidéo stockée dans un compartiment Amazon S3.

```
suspend fun startModerationDetection(  
    channel: NotificationChannel?,  
    bucketVal: String?,  
    videoVal: String?,  
) {  
    val s3obj =  
        S3Object {  
            bucket = bucketVal  
            name = videoVal  
        }  
    val vidObj =  
        Video {  
            s3object = s3obj  
        }  
    val request =  
        StartContentModerationRequest {  
            jobTag = "Moderation"  
            notificationChannel = channel  
            video = vidObj  
        }  
  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        val startModDetectionResult = rekClient.startContentModeration(request)  
        startJobId = startModDetectionResult.jobId.toString()  
    }  
}  
  
suspend fun getModResults() {  
    var finished = false  
    var status: String  
    var yy = 0  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        var modDetectionResponse: GetContentModerationResponse? = null  
  
        val modRequest =  
            GetContentModerationRequest {
```

```
        jobId = startJobId
        maxResults = 10
    }

    // Wait until the job succeeds.
    while (!finished) {
        modDetectionResponse = rekClient.getContentModeration(modRequest)
        status = modDetectionResponse.jobStatus.toString()
        if (status.compareTo("Succeeded") == 0) {
            finished = true
        } else {
            println("$yy status is: $status")
            delay(1000)
        }
        yy++
    }

    // Proceed when the job is done - otherwise VideoMetadata is null.
    val videoMetaData = modDetectionResponse?.videoMetadata
    println("Format: ${videoMetaData?.format}")
    println("Codec: ${videoMetaData?.codec}")
    println("Duration: ${videoMetaData?.durationMillis}")
    println("FrameRate: ${videoMetaData?.frameRate}")

    modDetectionResponse?.moderationLabels?.forEach { mod ->
        val seconds: Long = mod.timestamp / 1000
        print("Mod label: $seconds ")
        println(mod.moderationLabel)
    }
}
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la AWS Référence de l'API de SDK pour Kotlin.
 - [GetCelebrityRecognition](#)
 - [GetContentModeration](#)
 - [GetLabelDetection](#)
 - [GetPersonTracking](#)
 - [GetSegmentDetection](#)
 - [GetTextDetection](#)

- [StartCelebrityRecognition](#)
- [StartContentModeration](#)
- [StartLabelDetection](#)
- [StartPersonTracking](#)
- [StartSegmentDetection](#)
- [StartTextDetection](#)

Détecter des objets dans des images

L'exemple de code suivant montre comment créer une application qui utilise Amazon Rekognition pour détecter des objets par catégorie dans des images.

SDK pour Kotlin

Montre comment utiliser l'API Kotlin 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

Exemples d'enregistrement de domaine Route 53 à l'aide du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec enregistrement de domaine Route 53.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

Note

Il y en a plus 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 Kotlin 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-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <domainType>

        Where:
            domainType - The domain type (for example, com).
    """

    if (args.size != 1) {
        println(usage)
    }
}
```

```

        exitProcess(0)
    }

    val domainType = args[0]
    println("Invokes ListPrices using a Paginated method.")
    listPricesPaginated(domainType)
}

suspend fun listPricesPaginated(domainType: String) {
    val pricesRequest =
        ListPricesRequest {
            maxItems = 10
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
${pr.restorationPrice?.currency}")
            }
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [ListPrices](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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 Kotlin

Note

Il y en a plus 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 Kotlin 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-kotlin/latest/developer-guide/setup.html
```

```
This Kotlin 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.
*/

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
<lastName> <city>
        Where:
            domainType - The domain type (for example, com).
            phoneNumber - The phone number to use (for example, +1.2065550100)
            email - The email address to use.
            domainSuggestion - The domain suggestion (for example, findmy.example).
            firstName - The first name to use to register a domain.
            lastName - The last name to use to register a domain.
            city - The city to use to register a domain.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val domainType = args[0]
    val phoneNumber = args[1]
    val email = args[2]
    val domainSuggestion = args[3]
    val firstName = args[4]
    val lastName = args[5]
    val city = args[6]

    println(DASHES)
    println("Welcome to the Amazon Route 53 domains example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List current domains.")
    listDomains()
    println(DASHES)
}
```

```
println(DASHES)
println("2. List operations in the past year.")
listOperations()
println(DASHES)

println(DASHES)
println("3. View billing for the account in the past year.")
listBillingRecords()
println(DASHES)

println(DASHES)
println("4. View prices for domain types.")
listAllPrices(domainType)
println(DASHES)

println(DASHES)
println("5. Get domain suggestions.")
listDomainSuggestions(domainSuggestion)
println(DASHES)

println(DASHES)
println("6. Check domain availability.")
checkDomainAvailability(domainSuggestion)
println(DASHES)

println(DASHES)
println("7. Check domain transferability.")
checkDomainTransferability(domainSuggestion)
println(DASHES)

println(DASHES)
println("8. Request a domain registration.")
val opId = requestDomainRegistration(domainSuggestion, phoneNumber, email,
firstName, lastName, city)
println(DASHES)

println(DASHES)
println("9. Get operation details.")
getOperationalDetail(opId)
println(DASHES)

println(DASHES)
println("10. Get domain details.")
println("Note: You must have a registered domain to get details.")
```

```
println("Otherwise an exception is thrown that states ")
println("Domain xxxxxxxx not found in xxxxxxxx account.")
getDomainDetails(domainSuggestion)
println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
        ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
        ${response.registrantContact?.organizationName}")
    }
}

suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}

suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
        }
}
```

```
        countryCode = CountryCode.In
        email = emailVal
        firstName = firstNameVal
        lastName = lastNameVal
        city = cityVal
        phoneNumber = phoneNumberVal
        organizationName = "My Org"
        addressLine1 = "My Address"
        zipCode = "123 123"
    }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}

suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
            route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}

suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
}
```

```

    }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
        route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}

suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}

suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
                ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
                ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
                ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
                ${pr.restorationPrice?.currency}")
            }
    }
}

```



```

        }
    }
}

suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}

suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2

```

```
    }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}

suspend fun listDomains() {
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CheckDomainAvailability](#)
 - [CheckDomainTransferability](#)
 - [GetDomainDetail](#)
 - [GetDomainSuggestions](#)
 - [GetOperationDetail](#)
 - [ListDomains](#)
 - [ListOperations](#)
 - [ListPrices](#)
 - [RegisterDomain](#)
 - [ViewBilling](#)

Actions

CheckDomainAvailability

L'exemple de code suivant montre comment utiliser `CheckDomainAvailability`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
            route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CheckDomainAvailability](#) à la section AWS SDK pour la référence de l'API Kotlin.

CheckDomainTransferability

L'exemple de code suivant montre comment utiliser `CheckDomainTransferability`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
            route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CheckDomainTransferability](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetDomainDetail

L'exemple de code suivant montre comment utiliser `GetDomainDetail`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
        ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
        ${response.registrantContact?.organizationName}")
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [GetDomainDetail](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetDomainSuggestions

L'exemple de code suivant montre comment utiliser `GetDomainSuggestions`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetDomainSuggestions](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetOperationDetail

L'exemple de code suivant montre comment utiliser `GetOperationDetail`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetOperationDetail](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListDomains

L'exemple de code suivant montre comment utiliser `ListDomains`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listDomains() {
```

```
Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
    route53DomainsClient
        .listDomainsPaginated(ListDomainsRequest {})
        .transform { it.domains?.forEach { obj -> emit(obj) } }
        .collect { content ->
            println("The domain name is ${content.domainName}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDomains](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListOperations

L'exemple de code suivant montre comment utiliser `ListOperations`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
        currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(ListOperationsRequest {
                submittedSince = time2
            })
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The operations are: ${content.operations}")
            }
    }
}
```

```

        .listOperationsPaginated(operationsRequest)
        .transform { it.operations?.forEach { obj -> emit(obj) } }
        .collect { content ->
            println("Operation Id: ${content.operationId}")
            println("Status: ${content.status}")
            println("Date: ${content.submittedDate}")
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [ListOperations](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListPrices

L'exemple de code suivant montre comment utiliser `ListPrices`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
                ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
                ${pr.renewalPrice?.currency}")
            }
    }
}

```



```
        println("Transfer: ${pr.transferPrice?.price}
${pr.transferPrice?.currency}")
        println("Restoration: ${pr.restorationPrice?.price}
${pr.restorationPrice?.currency}")
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListPrices](#) à la section AWS SDK pour la référence de l'API Kotlin.

RegisterDomain

L'exemple de code suivant montre comment utiliser `RegisterDomain`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
        }
}
```

```
        phoneNumber = phoneNumberVal
        organizationName = "My Org"
        addressLine1 = "My Address"
        zipCode = "123 123"
    }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [RegisterDomain](#) à la section AWS SDK pour la référence de l'API Kotlin.

ViewBilling

L'exemple de code suivant montre comment utiliser `ViewBilling`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listBillingRecords() {
    val currentDate = Date()
```

```
    val localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ViewBilling](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples d'Amazon S3 utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon S3.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Principes de base](#)
- [Actions](#)
- [Scénarios](#)

Principes de base

Principes de base

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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun main(args: Array<String>) {  
    val 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).

"""

```
if (args.size != 4) {  
    println(usage)  
    exitProcess(1)  
}
```

```
val bucketName = args[0]  
val key = args[1]  
val objectPath = args[2]  
val savePath = args[3]  
val toBucket = args[4]
```

```
// Create an Amazon S3 bucket.  
createBucket(bucketName)
```

```
// Update a local file to the Amazon S3 bucket.  
putObject(bucketName, key, objectPath)
```

```
// Download the object to another local file.  
getObjectFromMrap(bucketName, key, savePath)
```

```
// List all objects located in the Amazon S3 bucket.  
listBucketObs(bucketName)
```

```
// Copy the object to another Amazon S3 bucket  
copyBucketOb(bucketName, key, toBucket)
```

```
// Delete the object from the Amazon S3 bucket.  
deleteBucketObs(bucketName, key)
```

```
// Delete the Amazon S3 bucket.  
deleteBucket(bucketName)  
println("All Amazon S3 operations were successfully performed")
```

```
}
```

```
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }
}
```

```
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
        }
}
```

```
        key = objectKey
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```


- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Actions

CopyObject

L'exemple de code suivant montre comment utiliser `CopyObject`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }
}
```

```
val request =
    CopyObjectRequest {
        copySource = encodedUrl
        bucket = toBucket
        key = objectKey
    }
S3Client { region = "us-east-1" }.use { s3 ->
    s3.copyObject(request)
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateBucket

L'exemple de code suivant montre comment utiliser `CreateBucket`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateMultiRegionAccessPoint

L'exemple de code suivant montre comment utiliser `CreateMultiRegionAccessPoint`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Configurez le client de contrôle S3 pour envoyer une demande à la région us-west-2.

```
suspend fun createS3ControlClient(): S3ControlClient {
    // Configure your S3ControlClient to send requests to US West (Oregon).
    val s3Control = S3ControlClient.fromEnvironment {
        region = "us-west-2"
    }
    return s3Control
}
```

Créez le point d'accès multirégional.

```
suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrapName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
```

```

        name = mrapName
        regions = listOf(
            Region {
                bucket = bucketName1
            },
            Region {
                bucket = bucketName2
            },
        )
    }
}

val requestToken: String? = createMrapResponse.requestTokenArn

// Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
if (requestToken != null) {
    waitForSucceededStatus(s3Control, requestToken, accountIdParam)
    println("MRAP created")
}

val getMrapResponse =
    s3Control.getMultiRegionAccessPoint(
        input = GetMultiRegionAccessPointRequest {
            accountId = accountIdParam
            name = mrapName
        },
    )
val mrapAlias = getMrapResponse.accessPoint?.alias
return "arn:aws:s3:::$accountIdParam:accesspoint/$mrapAlias"
}

```

Attendez que le point d'accès multirégional soit disponible.

```

suspend fun waitForSucceededStatus(
    s3Control: S3ControlClient,
    requestToken: String,
    accountIdParam: String,
    timeBetweenChecks: Duration = 1.minutes,
) {
    var describeResponse: DescribeMultiRegionAccessPointOperationResponse
    describeResponse = s3Control.describeMultiRegionAccessPointOperation(
        input = DescribeMultiRegionAccessPointOperationRequest {

```

```
        accountId = accountIdParam
        requestTokenArn = requestToken
    },
)

var status: String? = describeResponse.asyncOperation?.requestStatus
while (status != "SUCCEEDED") {
    delay(timeBetweenChecks)
    describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
        input = DescribeMultiRegionAccessPointOperationRequest {
            accountId = accountIdParam
            requestTokenArn = requestToken
        },
    )
    status = describeResponse.asyncOperation?.requestStatus
    println(status)
}
}
```

- Pour en savoir plus, consultez [Guide du développeur d'AWS SDK pour Kotlin](#).
- Pour plus de détails sur l'API, reportez-vous [CreateMultiRegionAccessPoint](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteBucketPolicy

L'exemple de code suivant montre comment utiliser DeleteBucketPolicy.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }
}
```

```
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteObjects

L'exemple de code suivant montre comment utiliser `DeleteObjects`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
```

```
        delete = delObj
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetBucketPolicy

L'exemple de code suivant montre comment utiliser `GetBucketPolicy`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetObject

L'exemple de code suivant montre comment utiliser `GetObject`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetObjectAcl

L'exemple de code suivant montre comment utiliser `GetObjectAcl`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectAcl](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListObjectsV2

L'exemple de code suivant montre comment utiliser `ListObjectsV2`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}


private fun calKb(intValue: Long): Long = intValue / 1024
```

- Pour plus de détails sur l'API, voir la [ListObjectsversion V2](#) dans le AWS SDK pour la référence de l'API Kotlin.

PutBucketAcl

L'exemple de code suivant montre comment utiliserPutBucketAcl.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAcl](#) à la section AWS SDK pour la référence de l'API Kotlin.

PutObject

L'exemple de code suivant montre comment utiliser PutObject.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section AWS SDK pour la référence de l'API Kotlin.

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 Kotlin

Note

Il y en a plus 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 demande présignée `GetObject` et utilisez l'URL pour télécharger un objet.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET request
    to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

Créez une demande `GetObject` présignée avec des options avancées.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be used
12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Créez une demande présignée `PutObject` et utilisez-la pour charger un objet.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)
```

```
// Use the URL and any headers from the presigned HttpRequest in a subsequent
// HTTP PUT request to retrieve the object.
// Create a PUT request using the OkHttpClient API.
val putRequest =
    Request
        .Builder()
        .url(presignedRequest.url.toString())
        .apply {
            presignedRequest.headers.forEach { key, values ->
                header(key, values.joinToString(", "))
            }
        }.put(content.toRequestBody())
        .build()

val response = OkHttpClient().newCall(putRequest).execute()
assert(response.isSuccessful)
}
```

- Pour en savoir plus, consultez [Guide du développeur d'AWS SDK pour Kotlin](#).

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

SDK pour Kotlin

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

Détecter des objets dans des images

L'exemple de code suivant montre comment créer une application qui utilise Amazon Rekognition pour détecter des objets par catégorie dans des images.

SDK pour Kotlin

Montre comment utiliser l'API Kotlin 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

Obtenir un objet depuis un point d'accès multirégional

L'exemple de code suivant montre comment obtenir un objet à partir d'un point d'accès multirégional.

SDK pour Kotlin

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Configurez le client S3 pour utiliser l'algorithme de signature asymétrique Sigv4 (Sigv4a).

```
suspend fun createS3Client(): S3Client {
    // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a) signing
    algorithm.
    val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
    val s3 = S3Client.fromEnvironment {
        authSchemes = listOf(sigV4AScheme)
    }
    return s3
}
```

Utilisez l'ARN du point d'accès multirégional au lieu d'un nom de compartiment pour récupérer l'objet.

```
suspend fun getObjectFromMrap(
    s3: S3Client,
    mrapArn: String,
    keyName: String,
): String? {
    val request = GetObjectRequest {
        bucket = mrapArn // Use the ARN instead of the bucket name for object
        operations.
        key = keyName
    }

    var stringObj: String? = null
    s3.getObject(request) { resp ->
        stringObj = resp.body?.decodeToString()
        if (stringObj != null) {
            println("Successfully read $keyName from $mrapArn")
        }
    }
    return stringObj
}
```

- Pour en savoir plus, consultez [Guide du développeur d'AWS SDK pour Kotlin](#).
- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section AWS SDK pour la référence de l'API Kotlin.

SageMaker Exemples d'IA utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec SageMaker IA.

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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour SageMaker AI

Les exemples de code suivants montrent comment commencer à utiliser l' SageMaker IA.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listBooks() {
    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
            sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
            response.notebookInstances?.forEach { item ->
                println("The notebook name is: ${item.notebookInstanceName}")
            }
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListNotebookInstances](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreatePipeline

L'exemple de code suivant montre comment utiliser `CreatePipeline`.

SDK pour Kotlin

Note

Il y en a plus 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.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
    }
    println(jsonObject)
}
```

```
// Create the pipeline.
val pipelineRequest = CreatePipelineRequest {
    pipelineDescription = "Kotlin SDK example pipeline"
    roleArn = roleArnVal
    pipelineName = pipelineNameVal
    pipelineDefinition = jsonObject.toString()
}

SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
    sageMakerClient.createPipeline(pipelineRequest)
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreatePipeline](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeletePipeline

L'exemple de code suivant montre comment utiliser `DeletePipeline`.

SDK pour Kotlin

Note

Il y en a plus 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.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [DeletePipeline](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribePipelineExecution

L'exemple de code suivant montre comment utiliser `DescribePipelineExecution`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribePipelineExecution](#) à la section AWS SDK pour la référence de l'API Kotlin.

StartPipelineExecution

L'exemple de code suivant montre comment utiliser `StartPipelineExecution`.

SDK pour Kotlin

Note

Il y en a plus 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.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
    pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
            "S3Data": {
                "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
            },
            "Type": "S3_DATA"
        },
    """
```

```
        "DocumentType": "CSV"
    }""
println(inputJSON)
val para3 = Parameter {
    name = "parameter_vej_input_config"
    value = inputJSON
}

// Create an ExportVectorEnrichmentJobOutputConfig object.
val jobS3Data = VectorEnrichmentJobS3Data {
    s3Uri = output
}

val outputConfig = ExportVectorEnrichmentJobOutputConfig {
    s3Data = jobS3Data
}

val gson4: String = gson.toJson(outputConfig)
val para4: Parameter = Parameter {
    name = "parameter_vej_export_config"
    value = gson4
}
println("parameter_vej_export_config:" + gson.toJson(outputConfig))

val para5JSON =
    "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
    \"Longitude\"},\"YAttributeName\":\"Latitude\"}"

val para5: Parameter = Parameter {
    name = "parameter_step_1_vej_config"
    value = para5JSON
}

parameters.add(para1)
parameters.add(para2)
parameters.add(para3)
parameters.add(para4)
parameters.add(para5)

val pipelineExecutionRequest = StartPipelineExecutionRequest {
    pipelineExecutionDescription = "Created using Kotlin SDK"
    pipelineExecutionDisplayName = "$pipelineName-example-execution"
    pipelineParameters = parameters
    pipelineName = pipelineNameVal
}
```

```
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
        sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StartPipelineExecution](#) à la section AWS SDK pour la référence de l'API Kotlin.

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, consultez [Créer et exécuter des SageMaker pipelines à l'aide AWS SDKs de Community.aws](#).

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
```



```
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
<queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
<pipelineName>

        Where:
            sageMakerRoleName - The name of the Amazon SageMaker role.
            lambdaRoleName - The name of the AWS Lambda role.
            functionName - The name of the AWS Lambda function (for
example,SageMakerExampleFunction).
            functionKey - The name of the Amazon S3 key name that represents the Lambda
function (for example, SageMakerLambda.zip).
            queueName - The name of the Amazon Simple Queue Service (Amazon SQS) queue.
            bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
bucket.
            bucketFunction - The name of the Amazon S3 bucket that contains the Lambda
ZIP file.
            lnglatData - The file location of the latlongtest.csv file required for this
use case.
            spatialPipelinePath - The file location of the GeoSpatialPipeline.json file
required for this use case.
            pipelineName - The name of the pipeline to create (for example, sagemaker-
sdk-example-pipeline).
        """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]

```

```
val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

println(DASHES)
println("Welcome to the Amazon SageMaker pipeline example scenario.")
println(
    """
        This example workflow will guide you through setting up and running an
        Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and an
        Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
        reverse geocode addresses in an input file and store the results in an
export file.
    """.trimIndent(),
)
println(DASHES)

println(DASHES)
println("First, we will set up the roles, functions, and queue needed by the
SageMaker pipeline.")
val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
val queueUrl = checkQueue(queueName, functionName)
println(DASHES)

println(DASHES)
println("Setting up bucket $bucketName")
if (!checkBucket(bucketName)) {
    setupBucket(bucketName)
    println("Put $lnglatData into $bucketName")
    val objectKey = "samplefiles/latlongtest.csv"
    putS3Object(bucketName, objectKey, lnglatData)
}
println(DASHES)

println(DASHES)
println("Now we can create and run our pipeline.")
setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn, pipelineName)
val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
println("The pipeline execution ARN value is $pipelineExecutionARN")
waitForPipelineExecution(pipelineExecutionARN)
println("Wait 30 secs to get output results $bucketName")
TimeUnit.SECONDS.sleep(30)
```

```

getOutputResults(bucketName)
println(DASHES)

println(DASHES)
println(
    ""
        The pipeline has completed. To view the pipeline and runs in SageMaker
Studio, follow these instructions:
        https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-studio.html
    "").trimIndent(),
)
println(DASHES)

println(DASHES)
println("Do you want to delete the AWS resources used in this Workflow? (y/n)")
val `in` = Scanner(System.`in`)
val delResources = `in`.nextLine()
if (delResources.compareTo("y") == 0) {
    println("Lets clean up the AWS resources. Wait 30 seconds")
    TimeUnit.SECONDS.sleep(30)
    deleteEventSourceMapping(functionName)
    deleteSQSQueue(queueName)
    listBucketObjects(bucketName)
    deleteBucket(bucketName)
    delLambdaFunction(functionName)
    deleteLambdaRole(lambdaRoleName)
    deleteSagemakerRole(sageMakerRoleName)
    deletePipeline(pipelineName)
} else {
    println("The AWS Resources were not deleted!")
}
println(DASHES)

println(DASHES)
println("SageMaker pipeline scenario is complete.")
println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }
}

```

```
SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
    sageMakerClient.deletePipeline(pipelineRequest)
    println("*** Successfully deleted $pipelineNameVal")
}
}

suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IAMClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IAMClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
    }
}
```

```
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun dellambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}
```

```
suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
        bucket = bucketNameVal
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
```

```
        eventSourceMapping = event.uuid.toString()
    }
}

val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
    uuid = eventSourceMapping
}

LambdaClient { region = "us-west-2" }.use { lambdaClient ->
    lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
    println("The event mapping is deleted!")
}

}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3objects: List<Object>? = response.contents
        if (s3objects != null) {
            for (`object` in s3objects) {
                if (bucketName != null) {
```

```

        readObject(bucketName, (`object`.key))
    }
}

suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
    pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
    }
}

```



```

        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
            "S3Data": {
                "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
            },
            "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
        \"Longitude\", \"YAttributeName\": \"Latitude\"}}\"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
    }

```

```

        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {

```

```

        pipelineDescription = "Kotlin SDK example pipeline"
        roleArn = roleArnVal
        pipelineName = pipelineNameVal
        pipelineDefinition = jsonObject.toString()
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.createPipeline(pipelineRequest)
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.headBucket(headBucketRequest)
        println("$bucketName exists")
    }
}

```

```

        return true
    }
} catch (e: S3Exception) {
    println("Bucket does not exist")
}
return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }

    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }

    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
successful")
    }
}
}

```

```

// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
    val queueAtt: MutableMap<String, String> = HashMap()
    queueAtt.put("DelaySeconds", "5")
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
{ queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a new
// queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
    }
}

```

```

        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
}

suspend fun createNewFunction(myFunctionName: String, s3BucketName: String, myS3Key:
String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->

```

```

        val response = lambdaClient.getFunction(functionRequest)
        functionArn = response.configuration?.functionArn.toString()
        println("$functionArn exists")
    }
} catch (e: LambdaException) {
    println(e.message + " A new function will be created")
    functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
myHandler, myRole)
}
return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val 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\""+
        "]" +
        "}"

val request = CreateRoleRequest {
    roleName = roleNameVal
    assumeRolePolicyDocument = assumeRolePolicy
    description = "Created using the AWS SDK for Kotlin"
}
IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
    val roleResult = iamClient.createRole(request)

    // Attach the policies to the role.
    for (policy in sageMakerRolePolicies) {
        val attachRequest = AttachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policy
        }
        iamClient.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.toString()
}
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)

```



```

        roleArn = response.role?.arn.toString()
        println(roleArn)
    }
} catch (e: IamException) {
    println(e.message + " A new role will be created")
    roleArn = createLambdaRole(roleNameVal)
}

return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val 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\"" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IAMClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
        }
    }
}

```

```

        iamClient.attachRolePolicy(attachRequest)
    }

    // Allow time for the role to be ready.
    TimeUnit.SECONDS.sleep(15)
    println("Role ready with ARN " + roleResult.role?.arn)
    return roleResult.role?.arn.toString()
}
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<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
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
        "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
}

```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CreatePipeline](#)
 - [DeletePipeline](#)
 - [DescribePipelineExecution](#)
 - [StartPipelineExecution](#)
 - [UpdatePipeline](#)

Exemples de Secrets Manager utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec 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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [GetSecretValue](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples d'Amazon SES utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon SES.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Scénarios](#)

Scénarios

Créer une application web pour suivre les données DynamoDB

L'exemple de code suivant montre comment créer une application Web qui suit les éléments de travail dans une table Amazon DynamoDB et utilise Amazon Simple Email Service (Amazon SES) pour envoyer des rapports.

SDK pour Kotlin

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 une application web pour suivre les données Amazon Redshift

L'exemple de code suivant montre comment créer une application Web qui suit et génère des rapports sur les éléments de travail à l'aide d'une base de données Amazon Redshift.

SDK pour Kotlin

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

L'exemple de code suivant montre comment créer une application Web qui suit les éléments de travail dans une base de données Amazon Aurora Serverless et utilise Amazon Simple Email Service (Amazon SES) pour envoyer des rapports.

SDK pour Kotlin

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](#).

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS

- Services de données Amazon RDS
- Amazon SES

Détecter des objets dans des images

L'exemple de code suivant montre comment créer une application qui utilise Amazon Rekognition pour détecter des objets par catégorie dans des images.

SDK pour Kotlin

Montre comment utiliser l'API Kotlin 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

Exemples d'Amazon SNS utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec 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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.


Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

 Note

Il y en a plus 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 aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
 * Before running this Kotlin 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-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTopics](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateTopic

L'exemple de code suivant montre comment utiliser `CreateTopic`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTopic](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteTopic

L'exemple de code suivant montre comment utiliser `DeleteTopic`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTopic](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetTopicAttributes

L'exemple de code suivant montre comment utiliser `GetTopicAttributes`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getSnsTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetTopicAttributes](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListSubscriptions

L'exemple de code suivant montre comment utiliser `ListSubscriptions`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listSnsSubscriptions() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListSubscriptions](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListTopics

L'exemple de code suivant montre comment utiliser `ListTopics`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listSNSTopics() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTopics](#) à la section AWS SDK pour la référence de l'API Kotlin.

Publish

L'exemple de code suivant montre comment utiliser `Publish`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section [Publier](#) de la référence du kit SDK AWS pour l'API Kotlin.

SetTopicAttributes

L'exemple de code suivant montre comment utiliser `SetTopicAttributes`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
        }
}
```

```
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SetTopicAttributes](#) à la section AWS SDK pour la référence de l'API Kotlin.

Subscribe

L'exemple de code suivant montre comment utiliser `Subscribe`.

SDK pour Kotlin

Note

Il y en a plus 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.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

```
    }
}
```

Abonne une fonction Lambda à une rubrique.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section [S'abonner](#) de la référence du kit SDK AWS pour l'API Kotlin.

TagResource

L'exemple de code suivant montre comment utiliser TagResource.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun addTopicTags(topicArn: String) {
```

```
val tag =
    Tag {
        key = "Team"
        value = "Development"
    }

val tag2 =
    Tag {
        key = "Environment"
        value = "Gamma"
    }

val tagList = mutableListOf<Tag>()
tagList.add(tag)
tagList.add(tag2)

val request =
    TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

SnsClient { region = "us-east-1" }.use { snsClient ->
    snsClient.tagResource(request)
    println("Tags have been added to $topicArn")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [TagResource](#) à la section AWS SDK pour la référence de l'API Kotlin.

Unsubscribe

L'exemple de code suivant montre comment utiliser `Unsubscribe`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section [Se désabonner](#) de la référence du kit SDK AWS pour l'API Kotlin.

Scénarios

Créer une application pour soumettre des données à une table DynamoDB

L'exemple de code suivant montre comment créer une application qui envoie des données à une table Amazon DynamoDB et vous avertit lorsqu'un utilisateur met à jour la table.

SDK pour Kotlin

Indique comment créer une application Android native qui envoie des données à l'aide de l'API Kotlin Amazon DynamoDB et envoie un message texte à l'aide de l'API Kotlin Amazon SNS.

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

Création d'une application Amazon SNS

L'exemple de code suivant montre comment créer une application dotée de fonctionnalités d'abonnement et de publication et traduisant des messages.

SDK pour Kotlin

Indique comment utiliser l'API Kotlin Amazon SNS pour créer une application 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 relatives à la création d'une application Web, consultez l'exemple complet sur [GitHub](#).

Pour obtenir le code source complet et les instructions sur la façon de créer une application Android native, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon Translate

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

SDK pour Kotlin

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

Publier un message texte SMS

L'exemple de code suivant montre comment publier des SMS à l'aide d'Amazon SNS.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Pour plus d'informations sur l'API, consultez la section [Publier](#) de la référence du kit SDK AWS pour l'API Kotlin.

Publier des messages dans des files d'attente

L'exemple de code suivant illustre comment :

- Créer une rubrique (FIFO ou non FIFO).
- Abonner plusieurs files d'attente à la rubrique avec la possibilité d'appliquer un filtre.
- Publier des messages dans la rubrique.
- Interroger les files d'attente pour les messages reçus.

SDK pour Kotlin

Note

Il y en a plus 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 aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
```

```
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

1. Gives the user three options to choose from.
2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
6. Subscribes to the SQS queue.
7. Publishes a message to the topic.
8. Displays the messages.
9. Deletes the received message.
10. Unsubscribes from the topic.
11. Deletes the SNS topic.
*/

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
```

```

println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
println(
    """
        In this scenario, you will create an SNS topic and subscribe an SQS
queue to the topic.
        You can select from several options for configuring the topic and
the subscriptions for the queue.
        You can then post to the topic and see the results in the queue.
    """.trimIndent(),
)
println(DASHES)

println(DASHES)
println(
    """
        SNS topics can be configured as FIFO (First-In-First-Out).
        FIFO topics deliver messages in order and support deduplication and
message filtering.
        Would you like to work with FIFO topics? (y/n)
    """.trimIndent(),
)
useFIFO = input.nextLine()
if (useFIFO.compareTo("y") == 0) {
    selectFIFO = true
    println("You have selected FIFO")
    println(
        """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html. """ ,
    )

    println("Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)")
    duplication = input.nextLine()
    if (duplication.compareTo("y") == 0) {
        println("Enter a group id value")
        groupId = input.nextLine()
    } else {

```

```
        println("Enter deduplication Id value")
        deduplicationID = input.nextLine()
        println("Enter a group id value")
        groupId = input.nextLine()
    }
}
println(DASHES)

println(DASHES)
println("2. Create a topic.")
println("Enter a name for your SNS topic.")
topicName = input.nextLine()
if (selectFIFO) {
    println("Because you have selected a FIFO topic, '.fifo' must be appended to
the topic name.")
    topicName = "$topicName.fifo"
    println("The name of the topic is $topicName")
    topicArn = createFIFO(topicName, duplication)
    println("The ARN of the FIFO topic is $topicArn")
} else {
    println("The name of the topic is $topicName")
    topicArn = createSNSTopic(topicName)
    println("The ARN of the non-FIFO topic is $topicArn")
}
println(DASHES)

println(DASHES)
println("3. Create an SQS queue.")
println("Enter a name for your SQS queue.")
sqsQueueName = input.nextLine()
if (selectFIFO) {
    sqsQueueName = "$sqsQueueName.fifo"
}
sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
println("The queue URL is $sqsQueueUrl")
println(DASHES)

println(DASHES)
println("4. Get the SQS queue ARN attribute.")
sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
println("The ARN of the new queue is $sqsQueueArn")
println(DASHES)

println(DASHES)
```

```

println("5. Attach an IAM policy to the queue.")
// Define the policy to use.
val policy = """{
  "Statement": [
    {
      "Effect": "Allow",
        "Principal": {
          "Service": "sns.amazonaws.com"
        },
      "Action": "sqs:SendMessage",
        "Resource": "$sqsQueueArn",
        "Condition": {
          "ArnEquals": {
            "aws:SourceArn": "$topicArn"
          }
        }
      }
    ]
  }"""
setQueueAttr(sqsQueueUrl, policy)
println(DASHES)

println(DASHES)
println("6. Subscribe to the SQS queue.")
if (selectFIFO) {
  println(
    """If you add a filter to this subscription, then only the filtered
    messages will be received in the queue.
    For information about message filtering, see https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html
    For this example, you can filter messages by a "tone" attribute."""
  )
  println("Would you like to filter messages for $sqsQueueName's subscription
  to the topic $topicName? (y/n)")
  val filterAns: String = input.nextLine()
  if (filterAns.compareTo("y") == 0) {
    var moreAns = false
    println("You can filter messages by using one or more of the following
    \"tone\" attributes.")
    println("1. cheerful")
    println("2. funny")
    println("3. serious")
    println("4. sincere")
    while (!moreAns) {

```

```

        println("Select a number or choose 0 to end.")
        val ans: String = input.nextLine()
        when (ans) {
            "1" -> filterList.add("cheerful")
            "2" -> filterList.add("funny")
            "3" -> filterList.add("serious")
            "4" -> filterList.add("sincere")
            else -> moreAns = true
        }
    }
}

subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
println(DASHES)

println(DASHES)
println("7. Publish a message to the topic.")
if (selectFIFO) {
    println("Would you like to add an attribute to this message? (y/n)")
    val msgAns: String = input.nextLine()
    if (msgAns.compareTo("y") == 0) {
        println("You can filter messages by one or more of the following \"tone
        \" attributes.")
        println("1. cheerful")
        println("2. funny")
        println("3. serious")
        println("4. sincere")
        println("Select a number or choose 0 to end.")
        val ans: String = input.nextLine()
        msgAttValue = when (ans) {
            "1" -> "cheerful"
            "2" -> "funny"
            "3" -> "serious"
            else -> "sincere"
        }
        println("Selected value is $msgAttValue")
    }
    println("Enter a message.")
    message = input.nextLine()
    pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
} else {
    println("Enter a message.")
    message = input.nextLine()
}

```



```
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
```

```
val request = DeleteTopicRequest {
    topicArn = topicArnVal
}

SnsClient { region = "us-east-1" }.use { snsClient ->
    snsClient.deleteTopic(request)
    println("$topicArnVal was deleted")
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }
}
```

```

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

```

```
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
```

```

        mapOf(msgAttValue to messAttr)
    if (duplication.compareTo("y") == 0) {
        val request = PublishRequest {
            message = messageVal
            messageGroupId = groupIdVal
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.publish(request)
            println(result.messageId.toString() + " Message sent.")
        }
    } else {
        // Create a publish request with the message and attributes.
        val request = PublishRequest {
            topicArn = topicArnVal
            message = messageVal
            messageDeduplicationId = deduplicationID
            messageGroupId = groupIdVal
            messageAttributes = mapAtt
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.publish(request)
            println(result.messageId.toString() + " Message sent.")
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->

```

```

        val result = snsClient.subscribe(request)
        println(
            "The queue " + queueArnVal + " has been subscribed to the topic " +
            topicArnVal + "\n" +
            "with the subscription ARN " + result.subscriptionArn,
        )
        return result.subscriptionArn
    }
} else {
    request = SubscribeRequest {
        protocol = "sqs"
        endpoint = queueArnVal
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println("The queue $queueArnVal has been subscribed to the topic
        $topicArnVal with the subscription ARN ${result.subscriptionArn}")

        val attributeNameVal = "FilterPolicy"
        val gson = Gson()
        val jsonString = "{\"tone\": []}"
        val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
        val toneArray = jsonObject.getAsJsonArray("tone")
        for (value: String? in filterList) {
            toneArray.add(JsonPrimitive(value))
        }

        val updatedJsonString: String = gson.toJson(jsonObject)
        println(updatedJsonString)
        val attRequest = SetSubscriptionAttributesRequest {
            subscriptionArn = result.subscriptionArn
            attributeName = attributeNameVal
            attributeValue = updatedJsonString
        }

        snsClient.setSubscriptionAttributes(attRequest)
        return result.subscriptionArn
    }
}
}

```

```

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal

```

```
        attributes = attrs
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")

        val urlRequest = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
        return getQueueUrlResponse.queueUrl
    }
} else {
    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val urlRequest = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
        return getQueueUrlResponse.queueUrl
    }
}
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}
```



```
suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publish](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Exemples Amazon SQS utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec 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.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

Note

Il y en a plus 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.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
```

```
SqsClient { region = "us-east-1" }.use { sqsClient ->
    sqsClient
        .listQueuesPaginated { }
        .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
        .collect { queue ->
            println("The Queue URL is $queue")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListQueues](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Actions](#)
- [Scénarios](#)

Actions

CreateQueue

L'exemple de code suivant montre comment utiliser CreateQueue.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }
}
```

```
SqsClient { region = "us-east-1" }.use { sqsClient ->
    sqsClient.createQueue(createQueueRequest)
    println("Get queue url")

    val getQueueUrlRequest =
        GetQueueUrlRequest {
            queueName = queueNameVal
        }

    val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
    return getQueueUrlResponse.queueUrl.toString()
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateQueue](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteMessage

L'exemple de code suivant montre comment utiliser `DeleteMessage`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}
```

```
    }  
  }  
  
  suspend fun deleteQueue(queueUrlVal: String) {  
    val request =  
      DeleteQueueRequest {  
        queueUrl = queueUrlVal  
      }  
  
    SqsClient { region = "us-east-1" }.use { sqsClient ->  
      sqsClient.deleteQueue(request)  
      println("$queueUrlVal was deleted!")  
    }  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteMessage](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteQueue

L'exemple de code suivant montre comment utiliser `DeleteQueue`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteMessages(queueUrlVal: String) {  
  println("Delete Messages from $queueUrlVal")  
  
  val purgeRequest =  
    PurgeQueueRequest {  
      queueUrl = queueUrlVal  
    }  
  
  SqsClient { region = "us-east-1" }.use { sqsClient ->  
    sqsClient.purgeQueue(purgeRequest)  
  }  
}
```

```
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteQueue](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListQueues

L'exemple de code suivant montre comment utiliser `ListQueues`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }
}
```

```
SqsClient { region = "us-east-1" }.use { sqsClient ->
    val response = sqsClient.listQueues(listQueuesRequest)
    response.queueUrls?.forEach { url ->
        println(url)
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListQueues](#) à la section AWS SDK pour la référence de l'API Kotlin.

ReceiveMessage

L'exemple de code suivant montre comment utiliser `ReceiveMessage`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ReceiveMessage](#) à la section AWS SDK pour la référence de l'API Kotlin.

SendMessage

L'exemple de code suivant montre comment utiliser `SendMessage`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
            messageBody = message
            delaySeconds = 10
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }
}
```



```
val msg2 =
    SendMessageBatchRequestEntry {
        id = "id2"
        messageBody = "Hello from msg 2"
    }

val sendMessageBatchRequest =
    SendMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = listOf(msg1, msg2)
    }

SqsClient { region = "us-east-1" }.use { sqsClient ->
    sqsClient.sendMessageBatch(sendMessageBatchRequest)
    println("Batch message were successfully sent.")
}
}
```

- Pour plus de détails sur l'API, reportez-vous [SendMessage](#) à la section AWS SDK pour la référence de l'API Kotlin.

Scénarios

Création d'une application de messagerie

L'exemple de code suivant montre comment créer une application de messagerie à l'aide d'Amazon SQS.

SDK pour Kotlin

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

Publier des messages dans des files d'attente

L'exemple de code suivant illustre comment :

- Créer une rubrique (FIFO ou non FIFO).
- Abonner plusieurs files d'attente à la rubrique avec la possibilité d'appliquer un filtre.
- Publier des messages dans la rubrique.
- Interroger les files d'attente pour les messages reçus.

SDK pour Kotlin

Note

Il y en a plus 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 aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
```

```
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

1. Gives the user three options to choose from.
2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
6. Subscribes to the SQS queue.
7. Publishes a message to the topic.
8. Displays the messages.
9. Deletes the received message.
10. Unsubscribes from the topic.
11. Deletes the SNS topic.
*/

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
```

```

println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
println(
    """
        In this scenario, you will create an SNS topic and subscribe an SQS
queue to the topic.
        You can select from several options for configuring the topic and
the subscriptions for the queue.
        You can then post to the topic and see the results in the queue.
    """).trimIndent(),
)
println(DASHES)

println(DASHES)
println(
    """
        SNS topics can be configured as FIFO (First-In-First-Out).
        FIFO topics deliver messages in order and support deduplication and
message filtering.
        Would you like to work with FIFO topics? (y/n)
    """).trimIndent(),
)
useFIFO = input.nextLine()
if (useFIFO.compareTo("y") == 0) {
    selectFIFO = true
    println("You have selected FIFO")
    println(
        """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html. """ ,
    )

    println("Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)")
    duplication = input.nextLine()
    if (duplication.compareTo("y") == 0) {
        println("Enter a group id value")
        groupId = input.nextLine()
    } else {

```

```
        println("Enter deduplication Id value")
        deduplicationID = input.nextLine()
        println("Enter a group id value")
        groupId = input.nextLine()
    }
}
println(DASHES)

println(DASHES)
println("2. Create a topic.")
println("Enter a name for your SNS topic.")
topicName = input.nextLine()
if (selectFIFO) {
    println("Because you have selected a FIFO topic, '.fifo' must be appended to
the topic name.")
    topicName = "$topicName.fifo"
    println("The name of the topic is $topicName")
    topicArn = createFIFO(topicName, duplication)
    println("The ARN of the FIFO topic is $topicArn")
} else {
    println("The name of the topic is $topicName")
    topicArn = createSNSTopic(topicName)
    println("The ARN of the non-FIFO topic is $topicArn")
}
println(DASHES)

println(DASHES)
println("3. Create an SQS queue.")
println("Enter a name for your SQS queue.")
sqsQueueName = input.nextLine()
if (selectFIFO) {
    sqsQueueName = "$sqsQueueName.fifo"
}
sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
println("The queue URL is $sqsQueueUrl")
println(DASHES)

println(DASHES)
println("4. Get the SQS queue ARN attribute.")
sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
println("The ARN of the new queue is $sqsQueueArn")
println(DASHES)

println(DASHES)
```

```

println("5. Attach an IAM policy to the queue.")
// Define the policy to use.
val policy = """{
  "Statement": [
    {
      "Effect": "Allow",
        "Principal": {
          "Service": "sns.amazonaws.com"
        },
      "Action": "sqs:SendMessage",
        "Resource": "$sqsQueueArn",
        "Condition": {
          "ArnEquals": {
            "aws:SourceArn": "$topicArn"
          }
        }
      }
    ]
  }"""
setQueueAttr(sqsQueueUrl, policy)
println(DASHES)

println(DASHES)
println("6. Subscribe to the SQS queue.")
if (selectFIFO) {
  println(
    """If you add a filter to this subscription, then only the filtered
    messages will be received in the queue.
    For information about message filtering, see https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html
    For this example, you can filter messages by a "tone" attribute."""
  )
  println("Would you like to filter messages for $sqsQueueName's subscription
  to the topic $topicName? (y/n)")
  val filterAns: String = input.nextLine()
  if (filterAns.compareTo("y") == 0) {
    var moreAns = false
    println("You can filter messages by using one or more of the following
    \"tone\" attributes.")
    println("1. cheerful")
    println("2. funny")
    println("3. serious")
    println("4. sincere")
    while (!moreAns) {

```

```

        println("Select a number or choose 0 to end.")
        val ans: String = input.nextLine()
        when (ans) {
            "1" -> filterList.add("cheerful")
            "2" -> filterList.add("funny")
            "3" -> filterList.add("serious")
            "4" -> filterList.add("sincere")
            else -> moreAns = true
        }
    }
}

subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
println(DASHES)

println(DASHES)
println("7. Publish a message to the topic.")
if (selectFIFO) {
    println("Would you like to add an attribute to this message? (y/n)")
    val msgAns: String = input.nextLine()
    if (msgAns.compareTo("y") == 0) {
        println("You can filter messages by one or more of the following \"tone
        \" attributes.")
        println("1. cheerful")
        println("2. funny")
        println("3. serious")
        println("4. sincere")
        println("Select a number or choose 0 to end.")
        val ans: String = input.nextLine()
        msgAttValue = when (ans) {
            "1" -> "cheerful"
            "2" -> "funny"
            "3" -> "serious"
            else -> "sincere"
        }
        println("Selected value is $msgAttValue")
    }
    println("Enter a message.")
    message = input.nextLine()
    pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
} else {
    println("Enter a message.")
    message = input.nextLine()
}

```

```
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
```



```
val request = DeleteTopicRequest {
    topicArn = topicArnVal
}

SnsClient { region = "us-east-1" }.use { snsClient ->
    snsClient.deleteTopic(request)
    println("$topicArnVal was deleted")
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }
}
```

```
val deleteMessageBatchRequest = DeleteMessageBatchRequest {
    queueUrl = queueUrlVal
    entries = entriesVal
}

SqsClient { region = "us-east-1" }.use { sqsClient ->
    sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
    println("The batch delete of messages was successful")
}
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

```
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
```

```

        mapOf(msgAttValue to messAttr)
    if (duplication.compareTo("y") == 0) {
        val request = PublishRequest {
            message = messageVal
            messageGroupId = groupIdVal
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.publish(request)
            println(result.messageId.toString() + " Message sent.")
        }
    } else {
        // Create a publish request with the message and attributes.
        val request = PublishRequest {
            topicArn = topicArnVal
            message = messageVal
            messageDeduplicationId = deduplicationID
            messageGroupId = groupIdVal
            messageAttributes = mapAtt
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.publish(request)
            println(result.messageId.toString() + " Message sent.")
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->

```

```
        val result = snsClient.subscribe(request)
        println(
            "The queue " + queueArnVal + " has been subscribed to the topic " +
            topicArnVal + "\n" +
            "with the subscription ARN " + result.subscriptionArn,
        )
        return result.subscriptionArn
    }
} else {
    request = SubscribeRequest {
        protocol = "sqs"
        endpoint = queueArnVal
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println("The queue $queueArnVal has been subscribed to the topic
        $topicArnVal with the subscription ARN ${result.subscriptionArn}")

        val attributeNameVal = "FilterPolicy"
        val gson = Gson()
        val jsonString = "{\"tone\": []}"
        val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
        val toneArray = jsonObject.getAsJsonArray("tone")
        for (value: String? in filterList) {
            toneArray.add(JsonPrimitive(value))
        }

        val updatedJsonString: String = gson.toJson(jsonObject)
        println(updatedJsonString)
        val attRequest = SetSubscriptionAttributesRequest {
            subscriptionArn = result.subscriptionArn
            attributeName = attributeNameVal
            attributeValue = updatedJsonString
        }

        snsClient.setSubscriptionAttributes(attRequest)
        return result.subscriptionArn
    }
}
}
```

```
suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }
    }
}
```

```
        attributes = attrs
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")

        val urlRequest = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
        return getQueueUrlResponse.queueUrl
    }
} else {
    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val urlRequest = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
        return getQueueUrlResponse.queueUrl
    }
}
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}
```

```
suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publish](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Exemples de Step Functions utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Step Functions.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez 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 Kotlin

Note

Il y en a plus 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 aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 * Before running this Kotlin 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-kotlin/latest/developer-guide/setup.html
 */
```

```
suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListStateMachines](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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 Kotlin

 Note

Il y en a plus 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 aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 * To run this code example, place the chat_sfn_state_machine.json file into your
 * project's resources folder.
 *
 * 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
 */
```

Before running this Kotlin 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-kotlin/latest/developer-guide/setup.html>

This Kotlin code example performs the following tasks:

1. List activities using a paginator.
2. List state machines using a paginator.
3. Creates an activity.
4. Creates a state machine.
5. Describes the state machine.
6. Starts execution of the state machine and interacts with it.
7. Describes the execution.
8. Deletes the activity.
9. Deletes the state machine.

*/

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main(args: Array<String>) {
```

```
    val 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.
```

```
        jsonFile - The location of the chat_sfn_state_machine.json file. You can  
located it in resources/sample_files.
```

```
    """
```

```
    if (args.size != 4) {
```

```
        println(usage)
```

```
        exitProcess(0)
```

```
    }
```

```
    val roleName = args[0]
```

```
    val activityName = args[1]
```

```
    val stateMachineName = args[2]
```

```
    val jsonFile = args[3]
```

```
val sc = Scanner(System.`in`)
var action = false

val polJSON = """{
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "",
    "Effect": "Allow",
    "Principal": {
      "Service": "states.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }
]
}"""

println(DASHES)
println("Welcome to the AWS Step Functions example scenario.")
println(DASHES)

println(DASHES)
println("1. List activities using a Paginator.")
listActivitesPagnator()
println(DASHES)

println(DASHES)
println("2. List state machines using a paginator.")
listStatemachinesPagnator()
println(DASHES)

println(DASHES)
println("3. Create a new activity.")
val activityArn = createActivity(activityName)
println("The ARN of the Activity is $activityArn")
println(DASHES)

// Get JSON to use for the state machine and place the activityArn value into
it.
val stream = GetStream()
val jsonString = stream.getStream(jsonFile)

// Modify the Resource node.
val objectMapper = ObjectMapper()
```

```
val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
activityArn)

// Convert the modified Java object back to a JSON string.
val stateDefinition = objectMapper.writeValueAsString(root)
println(stateDefinition)

println(DASHES)
println("4. Create a state machine.")
val roleARN = createIAMRole(roleName, polJSON)
val stateMachineArn = createMachine(roleARN, stateMachineName, stateDefinition)
println("The ARN of the state machine is $stateMachineArn")
println(DASHES)

println(DASHES)
println("5. Describe the state machine.")
describeStateMachine(stateMachineArn)
println("What should ChatSFN call you?")
val userName = sc.nextLine()
println("Hello $userName")
println(DASHES)

println(DASHES)
// The JSON to pass to the StartExecution call.
val executionJson = "{ \"name\" : \"$userName\" }"
println(executionJson)
println("6. Start execution of the state machine and interact with it.")
val runArn = startWorkflow(stateMachineArn, executionJson)
println("The ARN of the state machine execution is $runArn")
var myList: List<String>
while (!action) {
    myList = getActivityTask(activityArn)
    println("ChatSFN: " + myList[1])
    println("$userName please specify a value.")
    val myAction = sc.nextLine()
    if (myAction.compareTo("done") == 0) {
        action = true
    }
    println("You have selected $myAction")
    val taskJson = "{ \"action\" : \"$myAction\" }"
    println(taskJson)
    sendTaskSuccess(myList[0], taskJson)
}
```

```
println(DASHES)

println(DASHES)
println("7. Describe the execution.")
describeExe(runArn)
println(DASHES)

println(DASHES)
println("8. Delete the activity.")
deleteActivity(activityArn)
println(DASHES)

println(DASHES)
println("9. Delete the state machines.")
deleteMachine(stateMachineArn)
println(DASHES)

println(DASHES)
println("The AWS Step Functions example scenario is complete.")
println(DASHES)
}

suspend fun listStatemachinesPagnator() {
    val machineRequest =
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listStateMachinesPaginated(machineRequest)
            .transform { it.stateMachines?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The state machine ARN is ${obj.stateMachineArn}")
            }
    }
}

suspend fun listActivitesPagnator() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }
}
```

```
SfnClient { region = "us-east-1" }.use { sfnClient ->
    sfnClient
        .listActivitiesPaginated(activitiesRequest)
        .transform { it.activities?.forEach { obj -> emit(obj) } }
        .collect { obj ->
            println(" The activity ARN is ${obj.activityArn}")
        }
    }
}

suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}

suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}

suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient { region = "us-east-1" }.use { sfnClient ->
```



```
        val response = sfncClient.describeExecution(executionRequest)
        status = response.status.toString()
        if (status.compareTo("Running") == 0) {
            println("The state machine is still running, let's wait for it to
finish.")
            Thread.sleep(2000)
        } else if (status.compareTo("Succeeded") == 0) {
            println("The Step Function workflow has succeeded")
            hasSucceeded = true
        } else {
            println("The Status is $status")
        }
    }
}
println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient { region = "us-east-1" }.use { sfncClient ->
        sfncClient.sendTaskSuccess(successRequest)
    }
}

suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient { region = "us-east-1" }.use { sfncClient ->
        val response = sfncClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}

suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }
}
```

```
    }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}

suspend fun createIAMRole(
    roleNameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = roleNameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [CreateActivity](#)
 - [CreateStateMachine](#)

- [DeleteActivity](#)
- [DeleteStateMachine](#)
- [DescribeExecution](#)
- [DescribeStateMachine](#)
- [GetActivityTask](#)
- [ListActivities](#)
- [ListStateMachines](#)
- [SendTaskSuccess](#)
- [StartExecution](#)
- [StopExecution](#)

Actions

CreateActivity

L'exemple de code suivant montre comment utiliser `CreateActivity`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateActivity](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateStateMachine

L'exemple de code suivant montre comment utiliser `CreateStateMachine`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateStateMachine](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteActivity

L'exemple de code suivant montre comment utiliser DeleteActivity.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteActivity](#) à la section AWS SDK pour la référence de l'API Kotlin.

DeleteStateMachine

L'exemple de code suivant montre comment utiliser DeleteStateMachine.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteStateMachine](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeExecution

L'exemple de code suivant montre comment utiliser `DescribeExecution`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
```

```

        println("The state machine is still running, let's wait for it to
finish.")
        Thread.sleep(2000)
    } else if (status.compareTo("Succeeded") == 0) {
        println("The Step Function workflow has succeeded")
        hasSucceeded = true
    } else {
        println("The Status is $status")
    }
    }
    println("The Status is $status")
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeExecution](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeStateMachine

L'exemple de code suivant montre comment utiliser `DescribeStateMachine`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}

```



```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeStateMachine](#) à la section AWS SDK pour la référence de l'API Kotlin.

GetActivityTask

L'exemple de code suivant montre comment utiliser `GetActivityTask`.

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getActivityTask(actArn: String?): List<String> {  
    val myList: MutableList<String> = ArrayList()  
    val getActivityTaskRequest =  
        GetActivityTaskRequest {  
            activityArn = actArn  
        }  
    SfnClient { region = "us-east-1" }.use { sfnClient ->  
        val response = sfnClient.getActivityTask(getActivityTaskRequest)  
        myList.add(response.taskToken.toString())  
        myList.add(response.input.toString())  
        return myList  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetActivityTask](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListActivities

L'exemple de code suivant montre comment utiliser `ListActivities`.

SDK pour Kotlin

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }


    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listActivities(activitiesRequest)
        response.activities?.forEach { item ->
            println("The activity ARN is ${item.activityArn}")
            println("The activity name is ${item.name}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListActivities](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListExecutions

L'exemple de code suivant montre comment utiliser `ListExecutions`.

SDK pour Kotlin

 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](#).

```
suspend fun getExeHistory(exeARN: String?) {
```

```
val historyRequest =
    GetExecutionHistoryRequest {
        executionArn = exeARN
        maxResults = 10
    }

SfnClient { region = "us-east-1" }.use { sfnClient ->
    val response = sfnClient.getExecutionHistory(historyRequest)
    response.events?.forEach { event ->
        println("The event type is ${event.type}")
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListExecutions](#) à la section AWS SDK pour la référence de l'API Kotlin.

ListStateMachines

L'exemple de code suivant montre comment utiliser `ListStateMachines`.

SDK pour Kotlin

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 aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 * Before running this Kotlin 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-kotlin/latest/developer-guide/setup.html
 */
```

```
suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListStateMachines](#) à la section AWS SDK pour la référence de l'API Kotlin.

SendTaskSuccess

L'exemple de code suivant montre comment utiliser `SendTaskSuccess`.

SDK pour Kotlin

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](#).

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
```

```
        taskToken = token
        output = json
    }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SendTaskSuccess](#) à la section AWS SDK pour la référence de l'API Kotlin.

StartExecution

L'exemple de code suivant montre comment utiliser `StartExecution`.

SDK pour Kotlin

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](#).

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [StartExecution](#) à la section AWS SDK pour la référence de l'API Kotlin.

Support exemples d'utilisation du SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants en utilisant le AWS SDK pour Kotlin avec. Support

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

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.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Mise en route

Bonjour Support

Les exemples de code suivants montrent comment démarrer avec Support.

SDK pour Kotlin

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](#).

```
/**
```

```
Before running this Kotlin 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-kotlin/latest/developer-guide/setup.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 Kotlin example performs the following task:

1. Gets and displays available services.

```
*/  
  
suspend fun main() {  
    displaySomeServices()  
}  
  
// Return a List that contains a Service name and Category name.  
suspend fun displaySomeServices() {  
    val servicesRequest =  
        DescribeServicesRequest {  
            language = "en"  
        }  
  
    SupportClient { region = "us-west-2" }.use { supportClient ->  
        val response = supportClient.describeServices(servicesRequest)  
        println("Get the first 10 services")  
        var index = 1  
  
        response.services?.forEach { service ->  
            if (index == 11) {  
                return@forEach  
            }  
  
            println("The Service name is: " + service.name)  
  
            // Get the categories for this service.  
            service.categories?.forEach { cat ->  
                println("The category name is ${cat.name}")  
                index++  
            }  
        }  
    }  
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeServices](#) à la section AWS SDK pour la référence de l'API Kotlin.

Rubriques

- [Principes de base](#)
- [Actions](#)

Principes de base

Principes de base

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 Kotlin

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](#).

```
/**  
Before running this Kotlin 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-kotlin/latest/developer-guide/setup.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 Kotlin 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.
- */

```
suspend fun main(args: Array<String>) {
    val usage = """
Usage:
    <fileAttachment>
Where:
    fileAttachment - The file can be a simple saved .txt file to use as an
email attachment.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val fileAttachment = args[0]
    println("***** Welcome to the AWS Support case example scenario.")
    println("***** Step 1. Get and display available services.")
    val sevCatList = displayServices()

    println("***** Step 2. Get and display Support severity levels.")
    val sevLevel = displaySevLevels()
```

```
println("***** Step 3. Create a support case using the selected service,
category, and severity level.")
val caseIdVal = createSupportCase(sevCatList, sevLevel)
if (caseIdVal != null) {
    println("Support case $caseIdVal was successfully created!")
} else {
    println("A support case was not successfully created!")
    exitProcess(1)
}

println("***** Step 4. Get open support cases.")
getOpenCase()

println("***** Step 5. Create an attachment set with a generated file to add to
the case.")
val attachmentSetId = addAttachment(fileAttachment)
println("The Attachment Set id value is $attachmentSetId")

println("***** Step 6. Add communication with the attachment to the support
case.")
addAttachSupportCase(caseIdVal, attachmentSetId)

println("***** Step 7. List the communications of the support case.")
val attachId = listCommunications(caseIdVal)
println("The Attachment id value is $attachId")

println("***** Step 8. Describe the attachment set included with the
communication.")
describeAttachment(attachId)

println("***** Step 9. Resolve the support case.")
resolveSupportCase(caseIdVal)

println("***** Step 10. Get a list of resolved cases for the current day.")
getResolvedCase()
println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
```

```
val describeCasesRequest =
    DescribeCasesRequest {
        maxResults = 30
        afterTime = yesterday.toString()
        beforeTime = now.toString()
        includeResolvedCases = true
    }

SupportClient { region = "us-west-2" }.use { supportClient ->
    val response = supportClient.describeCases(describeCasesRequest)
    response.cases?.forEach { sinCase ->
        println("The case status is ${sinCase.status}")
        println("The case Id is ${sinCase.caseId}")
        println("The case subject is ${sinCase.subject}")
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
```

```
        caseId = caseIdVal
        maxResults = 10
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
case")
        } else {
            println("There was an error adding the communication to an AWS Support
case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
```

```
        fileName = myFile.name
        data = sourceBytes
    }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
```

```

        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

        SupportClient { region = "us-west-2" }.use { supportClient ->
            val response = supportClient.createCase(caseRequest)
            return response.caseId
        }
    }

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }
}

```

```
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Kotlin API reference.
 - [AddAttachmentsToSet](#)
 - [AddCommunicationToCase](#)
 - [CreateCase](#)
 - [DescribeAttachment](#)

- [DescribeCases](#)
- [DescribeCommunications](#)
- [DescribeServices](#)
- [DescribeSeverityLevels](#)
- [ResolveCase](#)

Actions

AddAttachmentsToSet

L'exemple de code suivant montre comment utiliser `AddAttachmentsToSet`.

SDK pour Kotlin

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](#).

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment)).readBytes()
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```


- Pour plus de détails sur l'API, reportez-vous [AddAttachmentsToSet](#) à la section AWS SDK pour la référence de l'API Kotlin.

AddCommunicationToCase

L'exemple de code suivant montre comment utiliser `AddCommunicationToCase`.

SDK pour Kotlin

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](#).

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
case")
        } else {
            println("There was an error adding the communication to an AWS Support
case")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [AddCommunicationToCase](#) à la section AWS SDK pour la référence de l'API Kotlin.

CreateCase

L'exemple de code suivant montre comment utiliser `CreateCase`.

SDK pour Kotlin

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](#).

```
suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateCase](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeAttachment

L'exemple de code suivant montre comment utiliser `DescribeAttachment`.

SDK pour Kotlin

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](#).

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeAttachment](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeCases

L'exemple de code suivant montre comment utiliser `DescribeCases`.

SDK pour Kotlin

 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](#).

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCases](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeCommunications

L'exemple de code suivant montre comment utiliser `DescribeCommunications`.

SDK pour Kotlin

 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](#).

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeCommunications](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeServices

L'exemple de code suivant montre comment utiliser `DescribeServices`.

SDK pour Kotlin

 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](#).

```
// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }
}
```

```
}

// Push the two values to the list.
serviceCode.let { sevCatList.add(it) }
catName.let { sevCatList.add(it) }
return sevCatList
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeServices](#) à la section AWS SDK pour la référence de l'API Kotlin.

DescribeSeverityLevels

L'exemple de code suivant montre comment utiliser `DescribeSeverityLevels`.

SDK pour Kotlin

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](#).

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeSeverityLevels](#) à la section AWS SDK pour la référence de l'API Kotlin.

ResolveCase

L'exemple de code suivant montre comment utiliser `ResolveCase`.

SDK pour Kotlin

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](#).

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ResolveCase](#) à la section AWS SDK pour la référence de l'API Kotlin.

Exemples d'Amazon Translate utilisant le SDK pour Kotlin

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour Kotlin avec Amazon Translate.

Les Scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la façon de configurer et d'exécuter le code en contexte.

Rubriques

- [Scénarios](#)

Scénarios

Création d'une application Amazon SNS

L'exemple de code suivant montre comment créer une application dotée de fonctionnalités d'abonnement et de publication et traduisant des messages.

SDK pour Kotlin

Indique comment utiliser l'API Kotlin Amazon SNS pour créer une application 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 relatives à la création d'une application Web, consultez l'exemple complet sur [GitHub](#).

Pour obtenir le code source complet et les instructions sur la façon de créer une application Android native, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon SNS
- Amazon Translate

Sécurité pour Kit AWS SDK pour Kotlin

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 Kit AWS SDK pour Kotlin](#)
- [Kit AWS SDK pour Kotlin support pour TLS 1.2](#)
- [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 Kit AWS SDK pour Kotlin

Le [modèle de responsabilité AWS partagée](#) de s'applique à la protection des données dans Kit AWS SDK pour Kotlin. 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. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section [Utilisation des CloudTrail sentiers](#) dans le guide de AWS CloudTrail l'utilisateur.
- 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 140-3 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 disponibles, consultez [Norme FIPS \(Federal Information Processing Standard\) 140-3](#).

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 Nom. Cela inclut lorsque vous travaillez avec le SDK pour Kotlin ou autre à Services AWS l'aide de la console, de l'API ou. AWS CLI AWS SDKs 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.

Kit AWS SDK pour Kotlin support pour TLS 1.2

Les informations suivantes s'appliquent uniquement à l'implémentation Java SSL (l'implémentation SSL par défaut dans le Kit AWS SDK pour Kotlin ciblage de la JVM). Si vous utilisez une implémentation de SSL différente, consultez votre implémentation de SSL spécifique pour savoir comment appliquer les versions de TLS.

Prise en charge de TLS dans Java

TLS 1.2 est pris en charge à partir de Java 7.

Comment vérifier la version de TLS

Pour vérifier quelle version de TLS est prise en charge dans votre machine virtuelle Java (JVM), vous pouvez utiliser le code suivant.

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.joinToString(separator = ", "))
```

Pour voir la liaison SSL en action et quelle version de TLS est utilisée, vous pouvez utiliser la propriété système `javax.net.debug`.

```
-Djavax.net.debug=ssl
```

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 des demandes vous-même, consultez [AWS Signature Version 4 pour les demandes 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 plus d'informations, consultez [Authentification multifactorielle](#) dans le Guide de l'utilisateur AWS IAM Identity Center et [Authentification multifactorielle AWS dans IAM](#) 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 racine, consultez [Tâches nécessitant des informations d'identification d'utilisateur racine](#) 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 une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme telles que des mots de passe et des 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 d'effectuer une rotation des 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 nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer les 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 plus d'informations, consultez [Cas d'utilisation pour les utilisateurs IAM](#) 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. Pour assumer temporairement un rôle IAM dans le AWS Management Console, vous

pouvez [passer d'un rôle d'utilisateur à un rôle IAM \(console\)](#). 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 [Méthodes pour endosser un rôle](#) 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 [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 [Accès intercompte aux ressources dans IAM](#) 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 une politique lors de la formulation de demandes FAS, consultez [Transmission des sessions 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 un rôle de 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 EC2 instance et qui envoient des demandes AWS CLI d' AWS API. Cela est préférable au stockage des clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour 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 exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez [Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon](#) 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 [Définition d'autorisations IAM personnalisées avec des politiques gérées par le client](#) 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. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. 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 (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et AWS WAF Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la [présentation de la liste 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 annule 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 (SCPs)** : SCPs 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 des politiques de contrôle des services (SCPs) 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 Organizations et consultez SCPs les [politiques de contrôle des services](#) dans le Guide de AWS Organizations l'utilisateur.

- **Politiques de contrôle des ressources (RCPs) :** RCPs politiques JSON que vous pouvez utiliser pour définir le maximum d'autorisations disponibles pour les ressources de vos comptes sans mettre à jour les politiques IAM associées à chaque ressource que vous possédez. Le RCP limite les autorisations pour les ressources des comptes membres et peut avoir un impact sur les autorisations effectives pour les identités, y compris Utilisateur racine d'un compte AWS, qu'elles appartiennent ou non à votre organisation. Pour plus d'informations sur les Organisations RCPs, y compris une liste de ces Services AWS supports RCPs, consultez la section [Resource control policies \(RCPs\)](#) dans le guide de AWS Organizations l'utilisateur.
- **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 session](#) 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 (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, 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 en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez [Accès intercompte aux ressources dans IAM](#) 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 :

- [Conformité et gouvernance de la sécurité](#) : ces guides de mise en œuvre de solutions traitent des considérations architecturales et fournissent les étapes à suivre afin de déployer des fonctionnalités de sécurité et de conformité.
- [Référence des services éligibles à la HIPAA — Répertoire les services éligibles](#) à la HIPAA. Tous ne Services AWS sont pas éligibles à la loi HIPAA.
- AWS Ressources de <https://aws.amazon.com/compliance/resources/> de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- [AWS Guides de conformité destinés aux clients](#) — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- [Évaluation des ressources à l'aide des règles](#) du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- [AWS Security Hub](#)— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez [Référence des contrôles Security Hub](#).
- [Amazon GuardDuty](#) — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.

- [AWS Audit Manager](#)— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Ce AWS produit ou service suit le [modèle de responsabilité partagée](#) par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la [AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité](#).

Résilience pour ce AWS produit ou service

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité.

Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant.

Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section [Infrastructure AWS mondiale](#).

Ce AWS produit ou service suit le [modèle de responsabilité partagée](#) par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la [AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité](#).

Sécurité de l'infrastructure pour ce AWS produit ou service

Ce AWS produit ou service utilise des services gérés et est donc protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section [Sécurité du AWS cloud](#). Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section [Protection de l'infrastructure](#) dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à ce AWS produit ou service via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser [AWS Security Token Service](#) (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Ce AWS produit ou service suit le [modèle de responsabilité partagée](#) par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la [AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité](#).

Historique du document

Cette rubrique décrit les modifications importantes apportées au Guide du Kit AWS SDK pour Kotlin développeur au cours de son histoire.

Modification	Description	Date
Protection de l'intégrité des données avec des checksums	Contenu mis à jour avec des détails sur le calcul automatique de la somme de contrôle.	16 janvier 2025
Mettre à jour le contenu de la chaîne de fournisseurs d'identifiants par	Chaîne de fournisseurs d'informations d'identification par défaut.	15 janvier 2025
Exemples de fichiers de construction de mises à jour	Afficher les éléments du fichier de construction tels que générés par Gradle version 8.11.1. Afficher l'utilisation de BOM. Intégrez des liens vers la dernière version des artefacts.	18 décembre 2024
Ajouter une rubrique DynamoDB Mapper (Developer Preview)	Associez des classes à des éléments DynamoDB à l'aide du mappeur DynamoDB (version préliminaire pour les développeurs)	29 octobre 2024
Mettre à jour les noms des compartiments Amazon S3	Sommes de contrôle Amazon S3 avec Kit AWS SDK pour Kotlin	30 septembre 2024
Ajouter des informations au moteur OkHttp 4	Spécifier un type de moteur HTTP	26 septembre 2024

Ajouter des informations sur les points de terminaison AWS basés sur un compte pour DynamoDB	Utiliser des points de AWS terminaison basés sur des comptes	24 septembre 2024
Ajouter une FAQ rubrique de résolution des problèmes	Dépannage FAQs	18 septembre 2024
Exemple OpenTelemetry de configuration de mise à jour et configuration du fournisseur de télémétrie global par défaut	Observabilité	2 mai 2024
Fournir plus de détails sur le processus de création du client de service	Création d'un client de service	14 mars 2024
Ajouter une rubrique sur les points d'accès multirégionaux	Travaillez avec les points d'accès multirégionaux Amazon S3 à l'aide du SDK pour Kotlin	6 février 2024
Ajouter les instructions du catalogue de versions Gradle	Catalogue des versions de Gradle (onglet)	19 décembre 2023
Version de disponibilité générale	Kit AWS SDK pour Kotlin Manuel du développeur	27 novembre 2023
Mettre à jour la section de configuration des points de terminaison du client en fonction des mises à jour du SDK	Points de terminaison clients	25 août 2023
Sommes de contrôle Amazon S3	Ajout d'une section expliquant comment utiliser des checksums flexibles avec Amazon S3.	14 août 2023

Ajouter une rubrique sur l'observabilité	Observabilité	3 août 2023
Ajouter une rubrique qui traite des nouvelles tentatives	Nouvelle tentative	7 juillet 2023
Mettre à jour la section de configuration du client HTTP en fonction des mises à jour du SDK	Configuration du client HTTP	6 juin 2023
Ajouter une rubrique de présignature HTTP	Demandes de présignature	2 juin 2023
Ajouter une rubrique sur les intercepteurs HTTP	Intercepteurs HTTP	22 mai 2023
Support pour l'actualisation automatique des jetons	Mettez à jour les instructions relatives à l'accès par authentification unique .	18 mai 2023
Sommes de contrôle Amazon S3	Ajoutez une section qui décrit comment utiliser les checksums avec Amazon S3 .	15 mai 2023
Remplacer la configuration du client de service	Ajoutez une section qui décrit comment remplacer la configuration d'un client de service et décrit comment les ressources sont affectées .	8 mai 2023
Appliquer une version minimale de TLS	Ajoutez une section qui décrit les options permettant d' appliquer une version minimale de TLS .	3 mai 2023
Configuration du point de terminaison client	Ajoutez une rubrique qui traite de la configuration du point de terminaison client .	7 avril 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 .	22 mars 2023
Ajouter un exemple de fichier de projet Maven	Montrez un exemple de fichier de projet Maven en plus d'un fichier de projet dans Gradle dans la rubrique Configuration .	2 décembre 2022
Réviser le contenu de la version préliminaire du guide du développeur	Contenu mis à jour pour refléter les récents travaux de développement	4 octobre 2022
Kit AWS SDK pour Kotlin Version préliminaire pour les développeurs	Kit AWS SDK pour Kotlin	2 décembre 2021
Kit AWS SDK pour Kotlin version alpha	Annonce de la nouvelle version Kit AWS SDK pour Kotlin alpha	30 août 2021

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.