

Guia do Desenvolvedor

AWS SDK para Kotlin



AWS SDK para Kotlin: Guia do Desenvolvedor

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

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é o AWS SDK para Kotlin?	1
Conceitos básicos da SDK	1
Manutenção e suporte para as versões SDK principais	1
Recursos adicionais	1
Conceitos básicos	3
Etapa 1: configurar para este tutorial	3
Etapa 2: criar o projeto	3
Etapa 3: escrever o código	6
Etapa 4: compilar e executar o aplicativo	8
Bem-sucedida	8
Limpeza	9
Próximas etapas	9
Configurar	10
Configuração básica	10
Visão geral	10
Capacidade de login no portal de acesso AWS	12
Configurar login único	12
Faça login usando o AWS CLI	13
Instale o Java e uma ferramenta de compilação	13
Usar credenciais temporárias	14
Crie arquivos de construção do projeto	15
Codifique seu projeto	20
Faça login usando o AWS CLI	21
Configurar	22
Crie um cliente de serviço	24
Configurar um cliente em código	24
Configurar um cliente a partir do ambiente	25
Feche o cliente	26
Região da AWS seleção	27
Cadeia de fornecedores da região padrão	27
Provedores de credenciais	28
Cadeia de fornecedores de credenciais padrão	28
Provedor de credenciais explícitas	31
Endpoints do cliente	32

Configuração personalizada	32
Exemplos	35
HTTP	36
Configuração do cliente HTTP	37
Usar um proxy HTTP	40
Interceptadores	41
Aplicar uma versão mínima do TLS	42
Repetições	44
Configuração padrão	44
Máximo de tentativas	44
Atrasos e recuos	45
Tente novamente o token bucket	48
Tentativas adaptáveis	51
Observabilidade	53
Configurar um TelemetryProvider	53
Métricas	55
Registro em log	57
Provedores de telemetria	61
Substituir a configuração do cliente	62
Ciclo de vida do cliente substituído	63
Recursos compartilhado	64
Usar a SDK	65
Fazer solicitações.	65
DSL Sobrecargas na interface de serviço	66
Solicitações sem entradas necessárias	67
Corrotinas	67
Fazendo solicitações simultâneas	67
Fazendo solicitações de bloqueio	68
Operações de streaming	69
Respostas de streaming	69
Solicitações de streaming	70
Paginação	71
Writers	72
Tratamento de erros	73
Exceções de serviço	73
Exceções do cliente	74

Metadados de erro	74
Solicitações pré-assinadas	75
Princípios básicos da pré-assinatura	75
Configuração avançada de pré-assinatura	76
Pré-assinatura POST e solicitações PUT	76
Operações que SDK podem preassinar	77
Solução de problemas do FAQs	78
Como faço para corrigir problemas de “conexão fechada”?	78
Por que as exceções são lançadas antes de atingir o máximo de tentativas?	79
Como faço para corrigir <code>NoSuchMethodError</code> ou <code>NoClassDefFoundError</code> ?	80
Trabalhe com Serviços da AWS	82
Amazon S3	83
Proteção da integridade de dados com somas de verificação	84
Trabalhe com pontos de acesso multirregionais	88
DynamoDB	94
Use AWS endpoints baseados em conta	94
Use o DynamoDB Mapper (Developer Preview)	95
Exemplos de código	122
API Gateway	123
Cenários	124
Aurora	124
Conceitos básicos	125
Ações	138
Cenários	124
Auto Scaling	152
Conceitos básicos	125
Ações	138
Amazon Bedrock	170
Ações	138
CloudWatch	171
Conceitos básicos	125
Ações	138
CloudWatch Registros	212
Ações	138
Provedor de identidade do Amazon Cognito	215
Ações	138

Cenários	124
Amazon Comprehend	231
Cenários	124
DynamoDB	232
Conceitos básicos	125
Ações	138
Cenários	124
Amazon EC2	262
Conceitos básicos	125
Ações	138
Amazon ECR	293
Conceitos básicos	125
Ações	138
OpenSearch Serviço	323
Ações	138
EventBridge	327
Conceitos básicos	125
Ações	138
AWS Glue	357
Conceitos básicos	125
Ações	138
IAM	369
Conceitos básicos	125
Ações	138
AWS IoT	388
Conceitos básicos	125
Ações	138
AWS IoT data	412
Ações	138
Amazon Keyspaces	414
Conceitos básicos	125
Ações	138
AWS KMS	439
Ações	138
Lambda	449
Conceitos básicos	125

Ações	138
Cenários	124
MediaConvert	458
Ações	138
Amazon Pinpoint	472
Ações	138
Amazon RDS	482
Conceitos básicos	125
Ações	138
Cenários	124
Serviços de dados do Amazon RDS	500
Cenários	124
Amazon Redshift	501
Ações	138
Cenários	124
Amazon Rekognition	505
Ações	138
Cenários	124
Registro de domínios do Route 53	524
Conceitos básicos	125
Ações	138
Amazon S3	543
Conceitos básicos	125
Ações	138
Cenários	124
SageMaker IA	565
Ações	138
Cenários	124
Secrets Manager	590
Ações	138
Amazon SES	591
Cenários	124
Amazon SNS	593
Ações	138
Cenários	124
Amazon SQS	621

Ações	138
Cenários	124
Step Functions	643
Conceitos básicos	125
Ações	138
Suporte	665
Conceitos básicos	125
Ações	138
Amazon Translate	683
Cenários	124
Segurança	685
Proteção de dados	685
Impor o TLS 1.2	687
Suporte para TLS em Java	687
Como verificar a versão do TLS	687
Gerenciamento de Identidade e Acesso	687
Público	688
Autenticar com identidades	688
Gerenciar o acesso usando políticas	692
Como Serviços da AWS trabalhar com o IAM	695
Solução de problemas AWS de identidade e acesso	695
Validação de conformidade	697
Resiliência	698
Segurança da infraestrutura	699
Histórico do documento	701
.....	dccv

O que é o AWS SDK para Kotlin?

AWS SDK para Kotlin Ele fornece Kotlin APIs para Amazon Web Services. Usando oSDK, você pode criar aplicativos Kotlin que funcionam com Amazon S3, Amazon, EC2 Amazon DynamoDB e muito mais. Com o KotlinSDK, você pode segmentar a JVM plataforma ou o Android de API nível 24 ou superior. Support para plataformas adicionais, como JavaScript Native, será lançado em versões futuras.

Para acompanhar os próximos recursos nos futuros lançamentos, consulte nosso [roteiro em GitHub](#).

Conceitos básicos da SDK

Para começar a usar oSDK, siga o [Conceitos básicos](#) tutorial.

Consulte [Configurar](#) para configurar seu ambiente de desenvolvimento.

Para criar e configurar clientes de serviço para fazer solicitações Serviços da AWS, consulte [Configuração](#). Para obter informações sobre vários recursos doSDK, consulte [Usar a SDK](#).

Para casos de uso e exemplos de execução de API operações específicas, consulte [Exemplos de código](#).

Manutenção e suporte para as versões SDK principais

Para obter informações sobre manutenção e suporte para as versões SDK principais e suas dependências subjacentes, consulte os tópicos a seguir no Guia de referência de ferramentas AWS SDKs e ferramentas:

- [AWS SDKsPolítica de manutenção de ferramentas e ferramentas](#)
- [AWS SDKsMatriz de suporte de versões e ferramentas](#)

Recursos adicionais

Além deste guia, a seguir estão recursos on-line valiosos SDK para desenvolvedores de Kotlin:

- [AWS blog do desenvolvedor](#)
- [Fóruns de desenvolvedores](#)

- [SDKfonte](#) (GitHub)
- [Catálogo de exemplos de código da AWS](#)
- [@awsdevelopers](#) (X, antigo Twitter)

Comece a usar o SDK para Kotlin

AWS SDK para Kotlin Ele fornece Kotlin APIs para cada um. AWS service (Serviço da AWS) Usando o SDK, você pode criar aplicativos Kotlin que funcionam com Amazon S3, Amazon, Amazon EC2 DynamoDB e muito mais.

Este tutorial mostra como usar o Gradle para definir dependências para o. AWS SDK para Kotlin Em seguida, você cria um código que grava dados em uma tabela do DynamoDB. Embora você queira usar os recursos de um IDE, tudo o que você precisa para este tutorial é uma janela de terminal e um editor de texto.

Siga estas etapas para concluir este tutorial:

- [Etapa 1: configurar para este tutorial](#)
- [Etapa 2: criar o projeto](#)
- [Etapa 3: escrever o código](#)
- [Etapa 4: compilar e executar o aplicativo](#)

Etapa 1: configurar para este tutorial

Antes de começar este tutorial, você precisa de um [conjunto de permissões do IAM Identity Center](#) que possa acessar o DynamoDB e de um ambiente de desenvolvimento Kotlin configurado com as configurações de login único do IAM Identity Center para acessar. AWS

Siga as instruções [Configuração básica](#) deste guia para obter a configuração básica deste tutorial.

Depois de configurar seu ambiente de desenvolvimento com [acesso de login único](#) para o Kotlin SDK e ter uma [sessão ativa do portal de AWS acesso](#), continue com a Etapa 2.

Etapa 2: criar o projeto

Para criar o projeto para este tutorial, primeiro use o Gradle para criar os arquivos básicos para um projeto Kotlin. Em seguida, atualize os arquivos com as configurações, dependências e código necessários para o. AWS SDK para Kotlin

Para criar um novo projeto usando o Gradle

Note

Este tutorial usa o Gradle versão 8.11.1 com o `gradle init` comando, que oferece cinco prompts na etapa 3 abaixo. Se você usa uma versão diferente do Gradle, os prompts podem ser diferentes, assim como as versões pré-preenchidas dos artefatos.

1. Crie um novo diretório chamado `getstarted` em um local de sua escolha, como sua área de trabalho ou pasta pessoal.
2. Abra uma janela de terminal ou prompt de comando e navegue até o `getstarted` diretório que você criou.
3. Use o comando a seguir para criar um novo projeto Gradle e uma classe básica do Kotlin.

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

- Quando o alvo for `solicitadoJava version`, pressione Enter (o padrão é). 21
- Quando `solicitadoProject name`, pressione Enter (o padrão é o nome do diretório, `getstarted` neste tutorial).
- Quando `solicitadoapplication structure`, pressione Enter (o padrão é). `Single application project`
- Quando `solicitadoSelect test framework`, pressione Enter (o padrão é). `kotlin.test`
- Quando `solicitadoGenerate build using new APIs and behavior`, pressione Enter (o padrão é). `no`

Para configurar seu projeto com dependências para o AWS SDK para Kotlin e o Amazon S3

- No `getstarted` diretório que você criou no procedimento anterior, substitua o conteúdo do `settings.gradle.kts` arquivo pelo conteúdo a seguir, `X.Y.Z` substituindo pela [versão mais recente](#) do SDK para 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")
```

- Navegue até o gradle diretório dentro do getstarted diretório. Substitua o conteúdo do arquivo de catálogo de versões nomeado `libs.versions.toml` pelo seguinte conteúdo:

```
[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" }
```

- Navegue até o diretório app e abra o arquivo `build.gradle.kts`. Substitua o conteúdo pelo código a seguir e salve as alterações.

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

dependencies {
    implementation(awssdk.services.s3) // Add dependency on the AWS SDK para 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()
}
```

A dependencies seção contém uma implementation entrada para o módulo Amazon S3 do AWS SDK para Kotlin O compilador Gradle está configurado para usar o Java 21 na java seção.

Etapa 3: escrever o código

Depois que o projeto for criado e configurado, edite a classe padrão do projeto App para usar o código de exemplo a seguir.

1. Na pasta do seu projetoapp, navegue até o diretóriosrc/main/kotlin/org/example. Abra o arquivo App.kt.
2. Substitua seu conteúdo pelo código a seguir e salve o arquivo.

```
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 = ByteStream.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!")
}
```

```
}
```

Etapa 4: compilar e executar o aplicativo

Depois que o projeto for criado e contiver a classe de exemplo, crie e execute o aplicativo.

1. Abra uma janela de terminal ou prompt de comando e navegue até o diretório `getstarted` do seu projeto.
2. Use o comando a seguir para criar e executar seu aplicativo:

```
gradle run
```

Note

Se você receber `umIdentityProviderException`, talvez não tenha uma sessão ativa de login único. Execute o comando `aws sso login` AWS CLI para iniciar uma nova sessão.

O aplicativo chama a operação da API [CreateBucket para criar um novo bucket](#) do S3 e, em seguida, chama o [PutObject](#) para colocar um novo objeto no novo bucket do S3.

Na `cleanup()` função no final, o aplicativo exclui o objeto e, em seguida, exclui o bucket do S3.

Para ver os resultados no console do Amazon S3

1. Em `App.kt`, comente a linha `cleanup(s3)` na `runBlocking` seção e salve o arquivo.
2. Reconstrua o projeto e coloque um novo objeto em um novo bucket do S3 executando `gradle run`
3. Faça login no [console do Amazon S3](#) para visualizar o novo objeto no novo bucket do S3.

Depois de visualizar o objeto, exclua o bucket do S3.

Bem-sucedida

Se seu projeto Gradle foi criado e executado sem erros, parabéns. Você construiu com sucesso seu primeiro aplicativo Kotlin usando o AWS SDK para Kotlin

Limpeza

Quando terminar de desenvolver seu novo aplicativo, exclua todos AWS os recursos que você criou durante este tutorial para evitar cobranças. Talvez você também queira excluir ou arquivar a pasta do projeto (get-started) que você criou na Etapa 2.

Siga estas etapas para limpar os recursos:

- Se você comentou a chamada para a `cleanup()` função, exclua o bucket do S3 usando o console do [Amazon S3](#).

Próximas etapas

Agora que você entendeu o básico, poderá descobrir mais sobre o seguinte:

- [Etapas adicionais de configuração para trabalhar com o SDK para Kotlin](#)
- [Configuração do SDK para Kotlin](#)
- [Usando o SDK para Kotlin](#)
- [Segurança para o SDK para Kotlin](#)

Configure o AWS SDK para Kotlin

Para fazer solicitações para Serviços da AWS usar o AWS SDK para Kotlin, você precisa do seguinte:

- A capacidade de fazer login no portal de acesso AWS
- Permissão para usar os AWS recursos de que seu aplicativo precisa
- Um ambiente de desenvolvimento com os seguintes elementos:
 - [Arquivos de configuração compartilhados](#) que são configurados com pelo menos uma das seguintes formas:
 - O config arquivo contém as configurações de credenciais do IAM Identity Center para que o SDK possa obter credenciais. AWS
 - O `credentials` arquivo contém credenciais temporárias
 - [Uma ferramenta de automação de construção, como Gradle ou Maven](#)
- Uma sessão ativa do portal de AWS acesso quando você estiver pronto para executar seu aplicativo

Neste tópico

- [Configuração básica](#)
- [Crie arquivos de construção do projeto](#)
- [Codifique seu projeto Kotlin usando o SDK para Kotlin](#)

Configuração básica

Visão geral

Para desenvolver com sucesso aplicativos que acessem Serviços da AWS usando o AWS SDK para Kotlin, os seguintes requisitos devem ser atendidos.

- Você deve conseguir [entrar no portal de acesso da AWS](#) disponível em AWS IAM Identity Center.
- As [permissões da função do IAM](#) configurada para o SDK devem permitir o acesso ao Serviços da AWS que seu aplicativo exige. As permissões associadas à política `PowerUserAccess AWS` gerenciada são suficientes para a maioria das necessidades de desenvolvimento.
- Um ambiente de desenvolvimento com os seguintes elementos:

- [Arquivos de configuração compartilhados](#) que são configurados de pelo menos uma das seguintes formas:
 - O arquivo config contém as [configurações de login único do Centro de Identidade do IAM](#) para que o SDK possa obter as credenciais da AWS .
 - O arquivo `credentials` contém credenciais temporárias.
- Uma [instalação do Java 8](#) ou posterior.
- Uma [ferramenta de automação de compilação](#), como [Maven](#) ou [Gradle](#).
- Um editor de texto para trabalhar com código.
- [\(Opcional, mas recomendado\) Um IDE \(ambiente de desenvolvimento integrado\), como IntelliJ IDEA ou Eclipse.](#)

Ao usar um IDE, você também pode integrar AWS Toolkit s para trabalhar com mais facilidade Serviços da AWS. O [AWS Toolkit para IntelliJ](#) e [AWS Toolkit for Eclipse](#) são dois kits de ferramentas que você pode usar.

- Uma sessão ativa do portal de AWS acesso quando você estiver pronto para executar seu aplicativo. Você usa o AWS Command Line Interface para [iniciar o processo de login no portal](#) de acesso do IAM Identity Center. AWS

Important

As instruções nesta seção de configuração pressupõem que você ou a organização usam o IAM Identity Center. Se sua organização usa um provedor de identidade externo que funciona independentemente do IAM Identity Center, descubra como você pode obter credenciais temporárias para o SDK para Kotlin usar. Siga estas instruções para adicionar credenciais temporárias ao `~/.aws/credentials` arquivo.

Se seu provedor de identidade adicionar credenciais temporárias automaticamente ao arquivo `~/.aws/credentials`, certifique-se de que o nome do perfil seja `[default]` para que você não precise fornecer um nome de perfil ao SDK ou à AWS CLI.

Capacidade de login no portal de acesso AWS

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

Se você não estiver familiarizado com o portal de AWS acesso, siga as orientações para acesso à conta no tópico de [autenticação do IAM Identity Center](#) no Guia de referência de ferramentas AWS SDKs e ferramentas.

Configurar acesso de logon único para o SDK

Depois de concluir a etapa 2 na [seção de acesso programático](#) para que o SDK use a autenticação do Centro de Identidade do IAM, o sistema deve conter os elementos a seguir.

- O AWS CLI, que você usa para iniciar uma [sessão do portal de AWS acesso](#) antes de executar seu aplicativo.
- Um arquivo `~/.aws/config` que contém um [perfil padrão](#). O SDK para Kotlin usa a configuração do provedor de token SSO do perfil para adquirir credenciais antes de enviar solicitações para AWS. O valor `sso_role_name`, que é um perfil do IAM conectado a um conjunto de permissões do Centro de Identidade do IAM, deve permitir o acesso a Serviços da AWS usado na aplicação.

O arquivo config de exemplo a seguir mostra um perfil padrão configurado com o provedor de token de SSO. A configuração `sso_session` do perfil se refere à seção chamada `sso-session`. A `sso-session` seção contém configurações para iniciar uma sessão do portal de AWS acesso.

```
[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
```

Para obter mais detalhes sobre as configurações usadas na configuração do provedor de token SSO, consulte Configuração do [provedor de token SSO](#) no Guia de referência AWS SDKs de ferramentas.

Se seu ambiente de desenvolvimento não estiver configurado para acesso programático conforme mostrado anteriormente, siga a [Etapa 2 no Guia de SDKs referência](#).

Faça login usando o AWS CLI

Antes de executar um aplicativo que acessa Serviços da AWS, você precisa de uma sessão ativa do portal de AWS acesso para que o SDK use a autenticação do IAM Identity Center para resolver as credenciais. Execute o comando a seguir no AWS CLI para entrar no portal de AWS acesso.

```
aws sso login
```

Como você tem uma configuração de perfil padrão, não precisa chamar o comando com uma `--profile` opção. Se a configuração do seu provedor de token SSO usa um perfil nomeado, o comando é `aws sso login --profile named-profile`.

Para testar se você já tem uma sessão ativa, execute o comando da AWS CLI a seguir.

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

A resposta a esse comando deve relatar a conta do Centro de Identidade do IAM e o conjunto de permissões configurados no arquivo compartilhado `config`.

Note

Se você já tiver uma sessão ativa do portal de acesso da AWS e executar `aws sso login`, não será necessário fornecer credenciais.

No entanto, você verá uma caixa de diálogo solicitando permissão para que o `botocore` acesse suas informações. O `botocore` é a base para a AWS CLI.

Selecione Permitir para autorizar o acesso às suas informações para o AWS CLI e o SDK para Kotlin.

Instale o Java e uma ferramenta de compilação

Seu ambiente de desenvolvimento necessita do seguinte:

- JDK 8 ou posterior. AWS SDK para Kotlin [Funciona com o Oracle Java SE Development Kit e com distribuições do Open Java Development Kit \(OpenJDK\) Amazon Corretto, como Red Hat OpenJDK e JDK. AdoptOpen](#)
- Uma ferramenta de compilação ou IDE compatível com o Maven Central, como Apache Maven, Gradle ou IntelliJ.
 - Para obter informações sobre como instalar e usar o Maven, consulte <http://maven.apache.org/>.
 - Para obter informações sobre como instalar e usar o Gradle, consulte <https://gradle.org/>.
 - Para obter informações sobre como instalar e usar o IntelliJ IDEA, consulte. <https://www.jetbrains.com/idea/>

Usar credenciais temporárias

Como alternativa à [configuração do acesso de login único do IAM Identity Center](#) para o SDK, você pode configurar seu ambiente de desenvolvimento com credenciais temporárias.

Configurar um arquivo de credenciais local para credenciais temporárias

1. [Criar um arquivo de credenciais compartilhadas](#)
2. No arquivo de credenciais, cole o seguinte texto de espaço reservado até colar as credenciais temporárias de trabalho:

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

3. Salve o arquivo. Agora, o arquivo `~/.aws/credentials` deve existir em seu sistema de desenvolvimento local. Esse arquivo contém o [perfil \[padrão\]](#) que o SDK para Kotlin usa se um perfil nomeado específico não for especificado.
4. [Faça login no portal de AWS acesso](#)
5. Siga essas instruções no título [Atualização manual de credenciais](#) para copiar as credenciais da função do IAM do AWS portal de acesso.
 - a. Na etapa 4 das instruções vinculadas, escolha o nome do perfil do IAM que concede acesso para suas necessidades de desenvolvimento. Essa função geralmente tem um nome como PowerUserAccess ou Desenvolvedor.

Para usar um catálogo de versões do Gradle

1. No seu `settings.gradle.kts` arquivo, adicione um `versionCatalogs` bloco dentro do `dependencyResolutionManagement` bloco.

O arquivo de exemplo a seguir configura o catálogo de AWS SDK para Kotlin versões. Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.

```
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. Declare dependências `build.gradle.kts` usando os identificadores de tipo seguro disponibilizados pelo catálogo de versões.

O arquivo de exemplo a seguir declara dependências para sete. Serviços da 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"
}
```

* Versão Java, por exemplo 17 ou 21.

Maven

O pom.xml arquivo de exemplo a seguir tem dependências para sete Serviços da AWS. Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.

```
<?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>
</dependencies>
```

```
<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>
```

* Versão Java, por exemplo 17 ou 21.

Codifique seu projeto Kotlin usando o SDK para Kotlin

Agora a diversão começa. Ao desenvolver seu aplicativo, você pode consultar a [Referência da AWS SDK para Kotlin API](#) para obter informações completas sobre as operações da API. Use os links a seguir para obter informações gerais da API Kotlin:

- [Referência da API de biblioteca padrão](#)
- [Visão geral das coroutines](#)
- [API de coroutines](#)

Faça login usando o AWS CLI

Sempre que você executa um programa que acessa Serviços da AWS, você precisa de uma sessão ativa do portal de AWS acesso. Você pode fazer isso executando o seguinte comando da :

```
aws sso login
```

Como você tem uma configuração de perfil padrão, não precisa chamar o comando com uma opção `--profile`. Se sua configuração de login único do IAM Identity Center usar um perfil nomeado, o comando será `aws sso login --profile named-profile`

Para testar se você já tem uma sessão ativa, execute o AWS CLI comando a seguir.

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

A resposta a esse comando deve relatar a conta do Centro de Identidade do IAM e o conjunto de permissões configurados no arquivo compartilhado `config`.

Configurar o AWS SDK para Kotlin

Esta seção explica como configurar um cliente de serviço usando AWS SDK para Kotlin. Para obter mais informações, consulte o [Guia de referência do SDK e das ferramentas](#), que inclui uma visão geral da configuração que se aplica a todos AWS SDKs.

Sumário

- [Crie um cliente de serviço](#)
 - [Configurar um cliente em código](#)
 - [Configurar um cliente a partir do ambiente](#)
 - [Feche o cliente](#)
- [Região da AWS seleção](#)
 - [Cadeia de fornecedores da região padrão](#)
- [Provedores de credenciais](#)
 - [A cadeia de fornecedores de credenciais padrão](#)
 - [Saiba mais sobre a cadeia de fornecedores de credenciais padrão](#)
 - [Provedor de credenciais explícitas](#)
- [Configurar endpoints do cliente](#)
 - [Configuração personalizada](#)
 - [Definir endpointUrl](#)
 - [Definir endpointProvider](#)
 - [Propriedades do EndpointProvider](#)
 - [endpointUrl ou endpointProvider](#)
 - [Uma observação sobre o Amazon S3](#)
 - [Exemplos](#)
 - [Exemplo de endpointUrl](#)
 - [Exemplo de endpointProvider](#)
 - [endpointUrl e endpointProvider](#)
- [HTTP](#)
 - [Configuração do cliente HTTP](#)
 - [Configuração básica](#)

- [Importações](#)
- [Código](#)
- [Especifique um tipo de mecanismo HTTP](#)
 - [Importações](#)
 - [Código](#)
 - [Usar a OkHttp4Engine](#)
 - [Use um cliente HTTP explícito](#)
 - [Importações](#)
 - [Código](#)
- [Usar um proxy HTTP](#)
 - [Use as propriedades do sistema JVM](#)
 - [Use variáveis de ambiente](#)
 - [Use um proxy nas EC2 instâncias](#)
- [Interceptores HTTP](#)
 - [Registro do interceptor](#)
 - [Interceptor para todas as operações do cliente de serviço](#)
 - [Interceptor somente para operações específicas](#)
- [Aplicar uma versão mínima do TLS](#)
 - [Configurar o mecanismo HTTP](#)
 - [Definir a propriedade do sdk.minTls sistema JVM](#)
 - [Defina a variável de SDK_MIN_TLS ambiente](#)
- [Repetições](#)
 - [Configuração de nova tentativa padrão](#)
 - [Máximo de tentativas](#)
 - [Atrasos e recuos](#)
 - [Tente novamente o token bucket](#)
 - [Tentativas adaptáveis](#)
- [Observabilidade](#)
 - [Configurar um TelemetryProvider](#)
 - [Configurar o provedor de telemetria global padrão](#)

- [Configurar um provedor de telemetria para um cliente de serviço específico](#)
- [Métricas](#)
- [Registro em log](#)
 - [Especifique o modo de registro para mensagens em nível de fio](#)
 - [Definir o modo de registro no código](#)
 - [Defina o modo de registro a partir do ambiente](#)
- [Provedores de telemetria](#)
 - [Configurar o provedor OpenTelemetry de telemetria baseado](#)
 - [Pré-requisitos](#)
 - [Configurar o SDK](#)
 - [Recursos](#)
- [Substituir a configuração do cliente de serviço](#)
- [Ciclo de vida de um cliente sobrecarregado](#)
- [Recursos compartilhados entre clientes](#)

Crie um cliente de serviço

Para fazer uma solicitação a um AWS service (Serviço da AWS), você deve primeiro instanciar um cliente para esse serviço.

Você pode definir configurações comuns para clientes de serviço, como o cliente HTTP a ser usado, o nível de registro e a configuração de repetição. Além disso, cada cliente de serviço exige um Região da AWS e um provedor de credenciais. O SDK usa esses valores para enviar solicitações para a região correta e assinar solicitações com as credenciais corretas.

Você pode especificar esses valores programaticamente no código ou fazer com que sejam carregados automaticamente do ambiente.

Configurar um cliente em código

Para configurar um cliente de serviço com valores específicos, você pode especificá-los em uma função lambda passada para o método de fábrica do cliente de serviço, conforme mostrado no trecho a seguir.

```
val dynamoDbClient = DynamoDbClient {
```



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

Todos os valores que você não especificar no bloco de configuração são definidos como padrões. Por exemplo, se você não especificar um provedor de credenciais como o código anterior, o provedor de credenciais usará como padrão a cadeia de fornecedores de credenciais [padrão](#).

Warning

Algumas propriedades, como `region` não têm um padrão. Você deve especificá-los explicitamente no bloco de configuração ao usar a configuração programática. Se o SDK não conseguir resolver a propriedade, as solicitações de API poderão falhar.

Configurar um cliente a partir do ambiente

Ao criar um cliente de serviço, o SDK pode inspecionar locais no ambiente de execução atual para determinar algumas propriedades de configuração. Esses locais incluem [arquivos compartilhados de configuração e credenciais](#), [variáveis de ambiente](#) e propriedades do sistema [JVM](#). As propriedades disponíveis para serem resolvidas incluem [AWS Região](#), [estratégia de repetição](#), [modo de registro](#) e outras. Para obter mais informações sobre todas as configurações que o SDK pode resolver no ambiente de execução, consulte o [Guia de referência de configurações AWS SDKs e ferramentas](#).

Para criar um cliente com configuração baseada no ambiente, use o método estático `suspend fun fromEnvironment()` na interface do cliente de serviço:

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

Criar um cliente dessa forma é útil quando executado na Amazon EC2 ou em qualquer outro contexto em que a configuração de um cliente de serviço esteja disponível no ambiente. AWS Lambda Isso separa seu código do ambiente em que ele está sendo executado e facilita a implantação do aplicativo em várias regiões sem alterar o código.

Além disso, você pode substituir propriedades específicas passando um bloco `lambda` para `fromEnvironment`. O exemplo a seguir carrega algumas propriedades de configuração do ambiente (por exemplo, Região), mas substitui especificamente o provedor de credenciais para usar credenciais de um perfil.

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

O SDK usa valores padrão para qualquer propriedade de configuração que não possa ser determinada a partir de configurações programáticas ou do ambiente. Por exemplo, se você não especificar um provedor de credenciais no código ou em uma configuração de ambiente, o provedor de credenciais usará como padrão a cadeia de fornecedores de credenciais [padrão](#).

Warning

Algumas propriedades, como Região, não têm um padrão. Você deve especificá-los em uma configuração de ambiente ou explicitamente no bloco de configuração. Se o SDK não conseguir resolver a propriedade, as solicitações de API poderão falhar.

Note

Embora propriedades relacionadas a credenciais, como chaves de acesso temporárias e configuração de SSO, possam ser encontradas no ambiente de execução, os valores não são fornecidos pelo cliente no momento da criação. Em vez disso, os valores são acessados pela camada do provedor de credenciais em cada solicitação.

Feche o cliente

Quando você não precisar mais do cliente de serviço, feche-o para liberar os recursos que ele está usando:

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

Como os clientes de serviço estendem a [Closeable](#) interface, você pode usar a [use](#) extensão para fechar o cliente automaticamente no final de um bloco, conforme mostrado no trecho a seguir.

```
DynamoDbClient.fromEnvironment().use { dynamoDbClient ->
```

```
// Invoke several DynamoDB operations.  
}
```

No exemplo anterior, o bloco lambda recebe uma referência ao cliente que acabou de ser criado. Você pode invocar operações nessa referência de cliente e, quando o bloco for concluído, inclusive lançando uma exceção, o cliente será fechado.

Região da AWS seleção

Com Regiões da AWS, você pode acessar Serviços da AWS essas operações em uma área geográfica específica. Isso pode ser útil para redundância e para manter os dados e os aplicativos em execução próximo ao lugar onde você e os usuários os acessarão.

Cadeia de fornecedores da região padrão

Ao carregar a configuração de um cliente de serviço [do ambiente](#), o seguinte processo de pesquisa é usado:

1. Qualquer região explícita definida no construtor.
2. A propriedade do sistema `aws.region` JVM é verificada. Se estiver definida, essa região será usada na configuração do cliente.
3. A variável de ambiente `AWS_REGION` está marcada. Se estiver definida, essa região será usada na configuração do cliente.
 - a. Observação: essa variável de ambiente é definida pelo contêiner Lambda.
4. O SDK verifica o arquivo de configuração AWS compartilhado. Se a `region` propriedade estiver definida para o perfil ativo, o SDK a usará.
 - a. A variável de ambiente `AWS_CONFIG_FILE` pode ser usada para personalizar o local do arquivo de configuração compartilhado.
 - b. A propriedade do sistema `aws.profile` JVM ou a variável de `AWS_PROFILE` ambiente podem ser usadas para personalizar o perfil que o SDK carrega.
5. O SDK tenta usar o Amazon EC2 Instance Metadata Service para determinar a região da instância atualmente em execução EC2 .
6. Se a região ainda não estiver resolvida neste momento, a criação do cliente falhará com uma exceção.

Provedores de credenciais

⚠ A ordem na qual a cadeia de provedores de credenciais padrão resolve as credenciais foi alterada com a versão 1.4.0. Para obter detalhes, consulte a nota abaixo.

Para fazer solicitações à Amazon Web Services usando o AWS SDK para Kotlin, o SDK usa credenciais assinadas criptograficamente emitidas por AWS. Para adquirir as credenciais, o SDK pode usar configurações que estão localizadas em vários lugares, por exemplo, propriedades do sistema JVM, variáveis de ambiente, arquivos compartilhados `AWS config` e metadados de `credentials` instâncias da Amazon. EC2

O SDK usa a abstração do provedor de credenciais para simplificar o processo de recuperação de credenciais de várias fontes. O SDK contém [várias implementações de provedores de credenciais](#).

Por exemplo, se a configuração recuperada incluir configurações de acesso de login único do IAM Identity Center do `config` arquivo compartilhado, o SDK trabalhará com o IAM Identity Center para recuperar as credenciais temporárias que ele usa para fazer solicitações. Serviços da AWS Com essa abordagem de aquisição de credenciais, o SDK usa o provedor do IAM Identity Center (também conhecido como provedor de credenciais de SSO). A [seção de configuração](#) deste guia descreveu essa configuração.

Para usar um provedor de credenciais específico, você pode especificar um ao criar um cliente de serviço. Como alternativa, você pode usar a cadeia de provedores de credenciais padrão para pesquisar as configurações automaticamente.

A cadeia de fornecedores de credenciais padrão

Quando não especificado explicitamente na construção do cliente, o SDK para Kotlin usa um provedor de credenciais que verifica sequencialmente cada local onde você pode fornecer credenciais. Esse provedor de credenciais padrão é implementado como uma cadeia de provedores de credenciais.

Para usar a cadeia padrão para fornecer credenciais em seu aplicativo, crie um cliente de serviço sem fornecer explicitamente uma `credentialsProvider` propriedade.

```
val ddb = DynamoDbClient {
```

```
    region = "us-east-2"  
}
```

Para obter mais informações sobre a criação de clientes de serviço, consulte [construir e configurar um cliente](#).

Saiba mais sobre a cadeia de fornecedores de credenciais padrão

A cadeia de provedores de credenciais padrão pesquisa a configuração de credenciais usando a seguinte sequência predefinida. Quando as configurações definidas fornecem credenciais válidas, a cadeia é interrompida.

1. [AWS chaves de acesso \(propriedades do sistema JVM\)](#)

O SDK procura as propriedades do sistema `aws.accessKeyId`, `aws.secretAccessKey`, e `aws.sessionToken` JVM.

2. [AWS chaves de acesso \(variáveis de ambiente\)](#)

O SDK tenta carregar credenciais das variáveis de ambiente `AWS_SESSION_TOKEN`, `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY`.

3. [Token de identidade da Web](#)

O SDK procura as variáveis de ambiente `AWS_WEB_IDENTITY_TOKEN_FILE` e `AWS_ROLE_ARN` (ou as propriedades `aws.webIdentityTokenFile` do sistema JVM e `aws.roleArn`). Com base nas informações do token e na função, o SDK adquire credenciais temporárias.

4. [Um perfil em um arquivo de configuração](#)

Nesta etapa, o SDK usa configurações associadas a um perfil. Por padrão, o SDK usa o compartilhado AWS config e os `credentials` arquivos, mas se a variável de ambiente `AWS_CONFIG_FILE` estiver definida, o SDK usará esse valor. Se a variável de ambiente `AWS_PROFILE` (ou propriedade do sistema `aws.profile` JVM) não estiver definida, o SDK procurará o perfil “padrão”, caso contrário, procurará o perfil que corresponde ao valor `AWS_PROFILE`.

O SDK procura o perfil com base na configuração descrita no parágrafo anterior e usa as configurações definidas lá. Se as configurações encontradas pelo SDK contiverem uma combinação de configurações para diferentes abordagens de provedores de credenciais, o SDK usará a seguinte ordem:

- a. [AWS chaves de acesso \(arquivo de configuração\)](#) - O SDK usa as configurações para `aws_access_key_id`, `aws_secret_access_key`, `aws_session_token`, e `aws_session_token`
- b. [Assumir a configuração da função](#) - se o SDK encontrar `role_arn`, `source_profile` e/ou `credential_source` configurações, ele tentará assumir uma função. Se o SDK encontrar a `source_profile` configuração, ele obterá credenciais de outro perfil para receber credenciais temporárias para a função especificada por `role_arn`. Se o SDK encontrar a `credential_source` configuração, ele obterá as credenciais de um contêiner do Amazon ECS, de uma EC2 instância da Amazon ou de variáveis de ambiente, dependendo do valor da configuração `credential_source`. Em seguida, ele usa essas credenciais para adquirir credenciais temporárias para a função.

Um perfil deve conter a `source_profile` configuração ou a `credential_source` configuração, mas não ambas.

- c. [Configuração do token de identidade da Web](#) - Se o SDK encontrar `role_arn` e `web_identity_token_file` configurar, ele adquire credenciais temporárias para acessar AWS recursos com base no `role_arn` e no token.
- d. [Configuração do token SSO](#) — se o SDK encontrar `sso_role_name` as configurações `sso_session`, `sso_account_id`, (junto com uma `sso-session` seção complementar nos arquivos de configuração), o SDK recuperará credenciais temporárias do serviço IAM Identity Center.
- e. [Configuração de SSO herdada](#) — se o SDK encontrar `sso_start_url`, `sso_region`, e `sso_role_name` definir `sso_account_id`, o SDK recuperará credenciais temporárias do serviço IAM Identity Center.
- f. [Configuração do processo](#): se o SDK encontrar uma `credential_process` configuração, ele usa o valor do caminho para invocar um processo e adquirir credenciais temporárias.

5. [Credenciais do contêiner](#)

O SDK procura variáveis de ambiente `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` ou `AWS_CONTAINER_CREDENTIALS_FULL_URI` e `AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE` ou `AWS_CONTAINER_AUTHORIZATION_TOKEN`. Ele usa esses valores para carregar credenciais do endpoint HTTP especificado por meio de uma solicitação GET.

6. [Credenciais IMDS](#)

O SDK tenta buscar credenciais do [Instance Metadata Service](#) no endpoint HTTP padrão ou configurado. O SDK só oferece suporte [IMDSv2](#).

Se as credenciais ainda não forem resolvidas neste momento, a criação do cliente falhará com uma exceção.

i Nota: Alteração na ordem de resolução de credenciais

A ordenação da resolução de credenciais descrita acima é atual para o 1.4.x+ lançamento do SDK para Kotlin. Antes do 1.4.0 lançamento, os itens número 3 e 4 foram trocados e o item 4a atual seguiu o item 4f atual.

Provedor de credenciais explícitas

Em vez de usar a cadeia de provedores padrão, você pode especificar um provedor de credenciais específico ou uma cadeia personalizada (`CredentialsProviderChain`) que o SDK deve usar. Por exemplo, se você definir as credenciais padrão usando variáveis de ambiente, forneça um `EnvironmentCredentialsProvider` para o construtor do cliente, como no trecho de código a seguir.

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

i Note

A cadeia padrão armazena em cache as credenciais, mas os provedores autônomos não. Você pode agrupar qualquer provedor de credenciais usando a `CachedCredentialsProvider` classe para evitar a busca desnecessária de credenciais em cada chamada de API. O provedor em cache só busca novas credenciais quando as atuais expiram.

i Note

Você pode implementar seu próprio provedor de credenciais ou cadeia de fornecedores implementando a `CredentialsProvider` interface.

Configurar endpoints do cliente

Quando ele AWS SDK para Kotlin chama um AWS service (Serviço da AWS), uma de suas primeiras etapas é determinar para onde encaminhar a solicitação. Esse processo é conhecido como resolução de endpoint.

Você pode configurar a resolução de endpoint para o SDK ao criar um cliente de serviço. A configuração padrão para resolução de endpoints geralmente é boa, mas há vários motivos que podem levar você a modificar a configuração padrão. Dois exemplos de motivos são os seguintes:

- Faça solicitações para uma versão de pré-lançamento de um serviço ou para uma implantação local de um serviço.
- Acesso a recursos de serviço específicos ainda não modelados no SDK.

Warning

A resolução de endpoints é um tópico avançado do SDK. Se você alterar as configurações padrão, corre o risco de quebrar seu código. As configurações padrão devem ser aplicadas à maioria dos usuários em ambientes de produção.

Configuração personalizada

Você pode personalizar a resolução do endpoint de um cliente de serviço com duas propriedades que estão disponíveis ao criar o cliente:

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

Definir `endpointUrl`

Você pode definir um valor `endpointUrl` para indicar um nome de host “base” para o serviço. Esse valor, no entanto, não é final, pois é passado como parâmetro para a `EndpointProvider` instância do cliente. A `EndpointProvider` implementação então pode inspecionar e potencialmente modificar esse valor para determinar o endpoint final.

Por exemplo, se você especificar um `endpointUrl` valor para um cliente do Amazon Simple Storage Service (Amazon S3) e realizar `GetObject` uma operação, a implementação padrão do provedor de endpoint injeta o nome do bucket no valor do nome do host.

Na prática, os usuários definem um `endpointUrl` valor para apontar para uma instância de desenvolvimento ou pré-visualização de um serviço.

Definir `EndpointProvider`

A `EndpointProvider` implementação de um cliente de serviço determina a resolução final do endpoint. A `EndpointProvider` interface mostrada no bloco de código a seguir expõe o `resolveEndpoint` método.

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

Um cliente de serviço chama o `resolveEndpoint` método para cada solicitação. O cliente do serviço usa o `Endpoint` valor retornado pelo provedor sem alterações adicionais.

Propriedades do `EndpointProvider`

O `resolveEndpoint` método aceita um `EndpointParameters` objeto específico do serviço que contém propriedades usadas na resolução de endpoints.

Cada serviço inclui as seguintes propriedades básicas.

Name	Tipo	Descrição
<code>region</code>	String	A AWS região do cliente
<code>endpoint</code>	String	Uma representação em cadeia do conjunto de valores de <code>endpointUrl</code>
<code>useFips</code>	Booleano	Se os endpoints FIPS estão habilitados na configuração do cliente

Name	Tipo	Descrição
useDualStack	Booleano	Se os endpoints de pilha dupla estão habilitados na configuração do cliente

Os serviços podem especificar propriedades adicionais necessárias para a resolução. Por exemplo, o Amazon S3 [S3EndpointParameters](#) inclui o nome do bucket e também várias configurações de recursos específicos do Amazon S3. Por exemplo, a `forcePathStyle` propriedade determina se o endereçamento do host virtual pode ser usado.

Se você implementar seu próprio provedor, não precisará criar sua própria instância do `EndpointParameters`. O SDK fornece as propriedades de cada solicitação e as passa para sua implementação do `resolveEndpoint`.

`endpointUrl` ou `endpointProvider`

É importante entender que as duas declarações a seguir NÃO produzem clientes com comportamento de resolução de endpoint equivalente:

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

// Use endpointProvider.
S3Client.fromEnvironment {
    endpointProvider = object : S3EndpointProvider {
        override suspend fun resolveEndpoint(params: S3EndpointParameters): Endpoint =
            Endpoint("https://endpoint.example")
    }
}
```

A declaração que define a `endpointUrl` propriedade especifica uma URL base que é passada para o provedor (padrão), que pode ser modificada como parte da resolução do endpoint.

A declaração que define o `endpointProvider` especifica o URL final que ele `S3Client` usa.

Embora você possa definir as duas propriedades, na maioria dos casos que precisam de personalização, você fornece uma delas. Como usuário geral do SDK, você geralmente fornece um `endpointUrl` valor.

Uma observação sobre o Amazon S3

O Amazon S3 é um serviço complexo com muitos de seus recursos modelados por meio de personalizações personalizadas de endpoints, como hospedagem virtual de bucket. A hospedagem virtual é um recurso do Amazon S3 em que o nome do bucket é inserido no nome do host.

Por isso, recomendamos que você não substitua a `EndpointProvider` implementação em um cliente de serviço do Amazon S3. Se você precisar estender seu comportamento de resolução, talvez enviando solicitações para uma pilha de desenvolvimento local com considerações adicionais de endpoint, recomendamos incluir a implementação padrão. O `endpointProvider` exemplo a seguir mostra um exemplo de implementação dessa abordagem.

Exemplos

Exemplo de `endpointUrl`

O trecho de código a seguir mostra como o endpoint de serviço geral pode ser substituído por um cliente Amazon S3.

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

Exemplo de `endpointProvider`

O trecho de código a seguir mostra como fornecer um provedor de endpoint personalizado que encapsule a implementação padrão para o 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 e endpointProvider

O programa de exemplo a seguir demonstra a interação entre as `endpointProvider` configurações `endpointUrl` e. Esse é um caso de uso avançado.

```
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

Esta seção aborda a configuração das configurações relacionadas ao HTTP no AWS SDK para Kotlin

Tópicos

- [Configuração do cliente HTTP](#)
- [Usar um proxy HTTP](#)
- [Interceptores HTTP](#)
- [Aplicar uma versão mínima do TLS](#)

Configuração do cliente HTTP

Por padrão, o AWS SDK para Kotlin usa um cliente HTTP baseado em [OkHttp](#). Você pode substituir o cliente HTTP e sua configuração fornecendo um cliente explicitamente configurado.

Note

Por padrão, cada cliente de serviço usa sua própria cópia de um cliente HTTP. Se você usa vários serviços em seu aplicativo, talvez queira criar um único cliente HTTP e compartilhá-lo com todos os clientes de serviço.

Configuração básica

Ao configurar um cliente de serviço, você pode configurar o tipo de mecanismo padrão. O SDK gerencia o mecanismo do cliente HTTP resultante e o fecha automaticamente quando não é mais necessário.

O exemplo a seguir mostra a configuração de um cliente HTTP durante a inicialização de um cliente do DynamoDB.

Importações

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

Código

```
DynamoDbClient {
    region = "us-east-2"
    httpClient {
```

```

        maxConcurrency = 64u
        connectTimeout = 10.seconds
    }
}.use { ddb ->

    // Perform some actions with Amazon DynamoDB.
}

```

Especifique um tipo de mecanismo HTTP

Para casos de uso mais avançados, você pode passar um parâmetro adicional `httpClient` que especifica o tipo de mecanismo. Dessa forma, você pode definir parâmetros de configuração exclusivos desse tipo de mecanismo.

O exemplo a seguir especifica o [OkHttpEngine](#) que você pode usar para configurar a [maxConcurrencyPerHost](#) propriedade.

Importações

```

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

```

Código

```

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.
}

```

Os valores possíveis para o tipo de motor são `OkHttpEngine`, [OkHttp4Engine](#), [CrtHttpEngine](#).

Para usar parâmetros de configuração específicos de um mecanismo HTTP, você deve adicionar o mecanismo como uma dependência em tempo de compilação. Para o `OkHttpClient`, você adiciona a seguinte dependência usando o Gradle.

(Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.)

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

Para o `CrtHttpClient`, adicione a seguinte dependência.

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

Usar a `OkHttpClient`

Use o `OkHttpClient` se você não puder usar o padrão `OkHttpClient`. O [GitHub repositório smithy-kotlin](#) tem informações sobre como você configura e usa o `OkHttpClient`

Use um cliente HTTP explícito

Ao usar um cliente HTTP explícito, você é responsável por sua vida útil, incluindo o fechamento quando não precisar mais dele. Um cliente HTTP deve durar pelo menos tanto quanto qualquer cliente de serviço que o use.

O exemplo de código a seguir mostra o código que mantém o cliente HTTP ativo enquanto o `DynamoDbClient` está ativo. A [use](#) função garante que o cliente HTTP feche corretamente.

Importações

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

Código

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

    DynamoDbClient {
```

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

Usar um proxy HTTP

Para acessar AWS por meio de servidores proxy usando o AWS SDK para Kotlin, você pode configurar as propriedades do sistema JVM ou as variáveis de ambiente. Se ambos forem fornecidos, as propriedades do sistema JVM terão precedência.

Use as propriedades do sistema JVM

O SDK procura as propriedades do sistema JVM `https.proxyHost`, e `https.proxyPort` `http.nonProxyHosts` Para obter mais informações sobre essas propriedades comuns do sistema JVM, consulte [Rede e proxies](#) na documentação do Java.

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

Use variáveis de ambiente

O SDK procura as variáveis de no_proxy ambiente `https_proxy``http_proxy`, e as versões em maiúsculas de cada uma.

```
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
```

Use um proxy nas EC2 instâncias

Se você configurar um proxy em uma EC2 instância executada com uma função do IAM anexada, certifique-se de isentar o endereço usado para acessar os [metadados da instância](#). Para fazer isso, defina a propriedade do sistema `http.nonProxyHosts` JVM ou a variável de `no_proxy` ambiente como o endereço IP do Serviço de Metadados da Instância, que é `169.254.169.254` Esse endereço não varia.


```
export no_proxy=169.254.169.254
```

Interceptores HTTP

Você pode usar interceptores para se conectar à execução de solicitações e respostas da API. Os interceptores são mecanismos abertos nos quais o SDK chama o código que você grava para injetar comportamento no ciclo de vida da solicitação/resposta. Dessa forma, você pode modificar uma solicitação em andamento, depurar o processamento da solicitação, visualizar exceções e muito mais.

O exemplo a seguir mostra um interceptor simples que adiciona um cabeçalho adicional a todas as solicitações de saída antes que o loop de repetição seja inserido.

```
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()  
    }  
}
```

Para obter mais informações e os ganchos de interceptação disponíveis, consulte a interface do [Interceptor](#).

Registro do interceptor

Você registra interceptores ao construir um cliente de serviço ou ao substituir a configuração de um conjunto específico de operações.

Interceptor para todas as operações do cliente de serviço

O código a seguir adiciona uma `AddHeader` instância à propriedade `interceptors` do construtor. Essa adição adiciona o `x-foo-version` cabeçalho a todas as operações antes que o loop de repetição seja inserido.

```
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 { ... }
```

Interceptor somente para operações específicas

Usando a `withConfig` extensão, você pode [substituir a configuração do cliente de serviço](#) para uma ou mais operações para qualquer cliente de serviço. Com esse recurso, você pode registrar interceptores adicionais para um subconjunto de operações.

O exemplo a seguir substitui a configuração da `s3` instância para operações dentro da `use` extensão. As operações chamadas `s3Scoped` contêm os cabeçalhos `x-foo-version` e os `x-bar-version` cabeçalhos.

```
// '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.
}
```

Aplicar uma versão mínima do TLS

Com o AWS SDK para Kotlin, você pode configurar a versão mínima do TLS ao se conectar aos endpoints de serviço. O SDK oferece diferentes opções de configuração. Na ordem da maior para a menor precedência, as opções são:

- Configurar explicitamente o mecanismo HTTP
- Definir a propriedade do `sdk.minTls` sistema JVM
- Defina a variável de `SDK_MIN_TLS` ambiente

Configurar o mecanismo HTTP

Ao especificar um mecanismo HTTP não padrão para um cliente de serviço, você pode definir o `tlsContext.minVersion` campo.

O exemplo a seguir configura o mecanismo HTTP e qualquer cliente de serviço que o usa para usar o TLS v1.2 no mínimo.

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

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

Definir a propriedade do **sdk.minTls** sistema JVM

Você pode definir a propriedade do sistema `sdk.minTls` JVM. Quando você inicia um aplicativo com a propriedade do sistema definida, todos os mecanismos HTTP criados pelo AWS SDK para Kotlin usam a versão mínima especificada do TLS por padrão. No entanto, você pode substituir isso explicitamente na configuração do mecanismo HTTP. Os valores permitidos são:

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

Defina a variável de **SDK_MIN_TLS** ambiente

Você pode definir a variável de `SDK_MIN_TLS` ambiente. Quando você inicia um aplicativo com a variável de ambiente definida, todos os mecanismos HTTP criados pelo AWS SDK para Kotlin usam a versão mínima especificada do TLS, a menos que sejam substituídos por outra opção.

Os valores permitidos são:

- TLS_1_0
- TLS_1_1
- TLS_1_2

- TLS_1_3

Repetições

Ligações para retornar Serviços da AWS ocasionalmente exceções inesperadas. Certos tipos de erros, como erros transitórios ou de limitação, podem ser bem-sucedidos se a chamada for repetida.

Esta página descreve como configurar novas tentativas automáticas com o AWS SDK para Kotlin

Configuração de nova tentativa padrão

Por padrão, cada cliente de serviço é configurado automaticamente com uma [estratégia de repetição padrão](#). A configuração padrão tenta uma chamada que falha até três vezes (a tentativa inicial mais duas tentativas). O atraso intermediário entre cada chamada é configurado com recuo exponencial e instabilidade aleatória para evitar tempestades de novas tentativas. Essa configuração funciona para a maioria dos casos de uso, mas pode ser inadequada em algumas circunstâncias, como sistemas de alto rendimento.

O SDK tenta novas tentativas somente em caso de erros que possam ser repetidos. Exemplos de erros que podem ser repetidos são tempos limite de soquete, limitação do lado do serviço, falhas de simultaneidade ou bloqueio otimistas e erros transitórios de serviço. Parâmetros ausentes ou inválidos, erros de autenticação/segurança e exceções de configuração incorreta não são considerados passíveis de nova tentativa.

Você pode personalizar a estratégia de repetição padrão definindo o máximo de tentativas, atrasos e recuos e a configuração do token bucket.

Máximo de tentativas

Você pode personalizar o máximo padrão de tentativas (3) no [bloco `retryStrategy` DSL](#) durante a construção do cliente.

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

Com o cliente de serviço do DynamoDB mostrado no snippet anterior, o SDK tenta chamadas de API que falham até cinco vezes (a tentativa inicial mais quatro tentativas).

Você pode desativar completamente as novas tentativas automáticas definindo o máximo de tentativas como uma, conforme mostrado no trecho a seguir.

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

Atrasos e recuos

Se uma nova tentativa for necessária, a estratégia de repetição padrão aguarda antes de fazer a tentativa subsequente. O atraso na primeira tentativa é pequeno, mas cresce exponencialmente nas tentativas posteriores. A quantidade máxima de atraso é limitada para que não cresça muito.

Finalmente, a instabilidade aleatória é aplicada aos atrasos entre todas as tentativas. A instabilidade ajuda a mitigar o efeito de grandes frotas que podem causar novas tempestades. (Veja esta [postagem no blog de AWS arquitetura](#) para uma discussão mais profunda sobre recuo exponencial e instabilidade.)


Os parâmetros de atraso são configuráveis no bloco [delayProviderDSL](#).

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

Com a configuração mostrada no trecho anterior, o cliente atrasa a primeira tentativa em até 100 milissegundos. O tempo máximo entre qualquer tentativa de repetição é de 5 segundos.

Os parâmetros a seguir estão disponíveis para atrasos de ajuste e recuo.

Parameter	Valor padrão	Descrição
<code>initialDelay</code>	10 milissegundos	A quantidade máxima de atraso para a primeira

Parameter	Valor padrão	Descrição
		tentativa. Quando a instabilidade é aplicada, a quantidade real de atraso pode ser menor.
<code> jitter </code>	1.0 (instabilidade total)	<p>A amplitude máxima pela qual reduzir aleatoriamente o atraso calculado. O valor padrão de 1,0 significa que o atraso calculado pode ser reduzido para qualquer valor de até 100% (por exemplo, até 0). Um valor de 0,5 significa que o atraso calculado pode ser reduzido em até metade. Assim, um atraso máximo de 10 ms pode ser reduzido para algo entre 5 ms e 10 ms. Um valor de 0,0 significa que nenhuma instabilidade é aplicada.</p> <div data-bbox="1068 1180 1507 1640" style="border: 1px solid #f08080; border-radius: 10px; padding: 10px;"><p> Important</p><p># A configuração do Jitter é um recurso avançado. A personalização desse comportamento normalmente não é recomendada.</p></div>

Parameter	Valor padrão	Descrição
maxBackoff	20 segundos	A quantidade máxima de atraso a ser aplicada a qualquer tentativa. Definir esse valor limita o crescimento exponencial que ocorre entre as tentativas subsequentes e evita que o máximo calculado seja muito grande. Esse parâmetro limita o atraso calculado antes que a instabilidade seja aplicada. Se aplicada, a instabilidade pode reduzir ainda mais o atraso.

Parameter	Valor padrão	Descrição
<code>scaleFactor</code>	1.5	<p>A base exponencial pela qual os atrasos máximos subsequentes serão aumentados. Por exemplo, considerando um <code>initialDelay</code> de 10 ms e um <code>scaleFactor</code> de 1,5, os seguintes atrasos máximos seriam calculados:</p> <ul style="list-style-type: none">• Tentativa 1 novamente: $10\text{ms} \times 1,5^0 = 10\text{ms}$• Tentativa 2 novamente: $10\text{ms} \times 1,5^1 = 15\text{ms}$• Tentativa 3 novamente: $10\text{ms} \times 1,5^2 = 22,5\text{ ms}$• Tentativa 4 novamente: $10\text{ ms} \times 1,5^3 = 33,75\text{ ms}$ <p>Quando a instabilidade é aplicada, a quantidade real de cada atraso pode ser menor.</p>

Tente novamente o token bucket

Você pode modificar ainda mais o comportamento da estratégia de repetição padrão ajustando a configuração padrão do token bucket. O repositório de tokens de repetição ajuda a reduzir as novas tentativas com menor probabilidade de sucesso ou que podem levar mais tempo para serem resolvidas, como falhas de tempo limite e limitação.

⚠ Important

A configuração do token bucket é um recurso avançado. A personalização desse comportamento normalmente não é recomendada.

Cada tentativa de nova tentativa (opcionalmente incluindo a tentativa inicial) diminui parte da capacidade do token bucket. O valor diminuído depende do tipo de tentativa. Por exemplo, repetir erros transitórios pode ser barato, mas tentar novamente erros de tempo limite ou de limitação pode ser mais caro.

Uma tentativa bem-sucedida retorna a capacidade para o bucket. O balde não pode ser incrementado além de sua capacidade máxima nem diminuído abaixo de zero.

Dependendo do valor da `useCircuitBreakerMode` configuração, as tentativas de diminuir a capacidade abaixo de zero resultam em um dos seguintes resultados:

- Se a configuração for `TRUE`, uma exceção será lançada — por exemplo, se muitas tentativas tiverem ocorrido e for improvável que mais tentativas sejam bem-sucedidas.
- Se a configuração for `FALSA`, haverá um atraso — por exemplo, atrasos até que o bucket tenha capacidade suficiente novamente.

Os parâmetros do token bucket são configuráveis no bloco [tokenBucketDSL](#):

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

Os parâmetros a seguir estão disponíveis para ajustar o repositório de tokens de repetição:

Parameter	Valor padrão	Descrição
<code>initialTryCost</code>	0	O valor a ser diminuído do intervalo nas tentativas

Parameter	Valor padrão	Descrição
		iniciais. O valor padrão de 0 significa que nenhuma capacidade será diminuída e, portanto, as tentativas iniciais não serão interrompidas ou atrasadas.
<code>initialTrySuccessIncrement</code>	1	O valor para incrementar a capacidade quando a tentativa inicial foi bem-sucedida.
<code>maxCapacity</code>	500	A capacidade máxima do token bucket. O número de tokens disponíveis não pode exceder esse número.
<code>refillUnitsPerSecond</code>	0	A quantidade de capacidade e adicionada novamente ao balde a cada segundo. Um valor de 0 significa que nenhuma capacidade é adicionada novamente automaticamente. (Por exemplo, somente tentativas bem-sucedidas resultam no aumento da capacidade). Um valor de 0 <code>useCircuitBreakerMode</code> precisa ser VERDADEIRO.

Parameter	Valor padrão	Descrição
<code>retryCost</code>	5	A quantidade a ser diminuída do compartimento em caso de uma tentativa após uma falha transitória. A mesma quantidade é reincrementada de volta para o bucket se a tentativa for bem-sucedida.
<code>timeoutRetryCost</code>	10	O valor a ser diminuído do bucket por uma tentativa após uma falha de tempo limite ou limitação. A mesma quantidade é reincrementada de volta para o bucket se a tentativa for bem-sucedida.
<code>useCircuitBreakerMode</code>	VERDADEIRO	Determina o comportamento quando uma tentativa de diminuir a capacidade faria com que a capacidade do bucket caísse abaixo de zero. Quando TRUE, o token bucket lançará uma exceção indicando que não existe mais capacidade de repetição. Quando FALSO, o token bucket atrasará a tentativa até que a capacidade suficiente seja reabastecida.

Tentativas adaptáveis

Como alternativa à estratégia de repetição padrão, a estratégia de repetição adaptativa é uma abordagem avançada que busca a taxa de solicitação ideal para minimizar os erros de limitação.

⚠ Important

As novas tentativas adaptáveis são um modo de repetição avançado. Normalmente, não é recomendável usar essa estratégia de repetição.

As novas tentativas adaptáveis incluem todos os recursos das novas tentativas padrão. Ele adiciona um limitador de taxa do lado do cliente que mede a taxa de solicitações limitadas em comparação com solicitações não limitadas. Também limita o tráfego para tentar permanecer dentro de uma largura de banda segura, o que, idealmente, causa zero erros de limitação.

A taxa se adapta em tempo real às mudanças nas condições de serviço e nos padrões de tráfego e pode aumentar ou diminuir a taxa de tráfego de acordo. Fundamentalmente, o limitador de taxa pode atrasar as tentativas iniciais em cenários de tráfego intenso.

Você seleciona a estratégia de repetição adaptativa fornecendo um parâmetro adicional ao `retryStrategy` método. Os parâmetros do limitador de taxa são configuráveis no bloco [rateLimiterDSL](#).

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

📘 Note

A estratégia de repetição adaptativa pressupõe que o cliente trabalhe com um único recurso (por exemplo, uma tabela do DynamoDB ou um bucket do Amazon S3). Se você usa um único cliente para vários recursos, a limitação ou as interrupções associadas a um recurso resultam em maior latência e falhas quando o cliente acessa todos os outros recursos. Ao usar a estratégia de repetição adaptável, recomendamos que você use um único cliente para cada recurso.

Observabilidade

Observabilidade é até que ponto o estado atual de um sistema pode ser inferido a partir dos dados que ele emite. Os dados emitidos são comumente chamados de telemetria.

Eles AWS SDK para Kotlin podem fornecer todos os três sinais comuns de telemetria: métricas, rastreamentos e registros. Você pode conectar um [TelemetryProvider](#) para enviar dados de telemetria para um back-end de observabilidade (como a [AWS X-RayAmazon CloudWatch](#)) e, em seguida, agir de acordo com eles.

Por padrão, somente o registro está ativado e outros sinais de telemetria são desativados no SDK. Este tópico explica como habilitar e configurar a saída de telemetria.

Important

`TelemetryProvider` atualmente é uma API experimental que deve ser ativada para uso.

Configurar um `TelemetryProvider`

Você pode configurar um `TelemetryProvider` em seu aplicativo globalmente para todos os clientes de serviço ou para clientes individuais. Os exemplos a seguir usam uma `getConfiguredProvider()` função hipotética para demonstrar as operações da `TelemetryProvider` API. A [the section called “Provedores de telemetria”](#) seção descreve as informações para implementações fornecidas pelo SDK. Se um provedor não for compatível, você poderá implementar seu próprio suporte ou [abrir uma solicitação de recurso no GitHub](#).

Configurar o provedor de telemetria global padrão

Por padrão, todo cliente de serviço tenta usar o provedor de telemetria disponível globalmente. Dessa forma, você pode definir o provedor uma vez e todos os clientes o usarão. Isso deve ser feito apenas uma vez, antes de você instanciar qualquer cliente de serviço.

Para usar o provedor global de telemetria, primeiro atualize as dependências do projeto para adicionar o módulo de padrões de telemetria, conforme mostrado no seguinte trecho do Gradle.

(Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.)

```
dependencies {
```

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

Em seguida, defina o provedor de telemetria global antes de criar um cliente de serviço, conforme mostrado no código a seguir.

```
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")
}
```

Configurar um provedor de telemetria para um cliente de serviço específico

Você pode configurar um cliente de serviço individual com um provedor de telemetria específico (diferente do global). Isso é mostrado no exemplo a seguir.

```
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")
}

```

Métricas

A tabela a seguir lista as métricas de telemetria que o SDK emite. [Configure um provedor de telemetria](#) para tornar as métricas observáveis.

Quais métricas são emitidas?

Nome da métrica	Unidades	Tipo	Atributos	Descrição
smithy.client.call.duration	s	Histograma	rpc.service, rpc.method	Duração geral da chamada (incluindo novas tentativas)
smithy.client.call.attempts	{tentativa}	Monoton Counter	rpc.service, rpc.method	O número de tentativas para uma operação individual
smithy.client.call.errors	{erro}	Monoton Counter	rpc.service, rpc.method, exceção.type	O número de erros de uma operação
smithy.client.call.tempt_duration	s	Histograma	rpc.service, rpc.method	O tempo necessário para se conectar ao serviço, enviar a solicitação e recuperar o código de status e os cabeçalhos HTTP (incluindo o tempo na fila de espera para ser enviado)
smithy.client.call.resolve_endpoint_duration	s	Histograma	rpc.service, rpc.method	O tempo necessário para resolver um endpoint (resolvedor de endpoint, não DNS) para a solicitação
smithy.client.call.serialization_duration	s	Histograma	rpc.service, rpc.method	O tempo necessário para serializar o corpo da mensagem
smithy.client.call.duração_desserialização	s	Histograma	rpc.service, rpc.method	O tempo necessário para desserializar o corpo da mensagem

Nome da métrica	Unidades	Tipo	Atributos	Descrição
smithy.client.auth.signing_duration	s	Histograma	rpc.service, rpc.method, auth.scheme_id	O tempo necessário para assinar uma solicitação
smithy.client.auth.resolve_identity_duration	s	Histograma	rpc.service, rpc.method, auth.scheme_id	O tempo necessário para adquirir uma identidade (como AWS credenciais ou um token de portador) de um provedor de identidade
smithy.client.http.connections.acquire_duration	s	Histograma		O tempo necessário para que uma solicitação adquira uma conexão
smithy.client.http.connections.limit	{conexão}	[Assíncrono] UpDownCounter		O máximo de conexões abertas permitidas/configuradas para o cliente HTTP
smithy.client.http.connections.usage	{conexão}	[Assíncrono] UpDownCounter	estado: ocioso adquirido	Estado atual do pool de conexões
smithy.client.http.connections.uptime	s	Histograma		A quantidade de tempo em que uma conexão foi aberta
smithy.client.http.requests.usage	{solicitação}	[Assíncrono] UpDownCounter	estado: em fila em voo	O estado atual da simultaneidade de solicitações do cliente HTTP

Nome da métrica	Unidades	Tipo	Atributos	Descrição
smithy.client.http.requests.queued_duration	s	Histograma		A quantidade de tempo que uma solicitação passou na fila e esperando para ser executada pelo cliente HTTP
smithy.client.http.bytes_sent	Por	Monoton Counter	endereço do servidor	O número total de bytes enviados pelo cliente HTTP
smithy.client.http.bytes_received	Por	Monoton Counter	endereço do servidor	O número total de bytes recebidos pelo cliente HTTP

A seguir estão as descrições das colunas:

- Nome da métrica — O nome da métrica emitida.
- Unidades — A unidade de medida da métrica. As unidades são fornecidas na notação [UCUM](#) com distinção entre maiúsculas e minúsculas (“c/s”).
- Tipo — O tipo de instrumento usado para capturar a métrica.
- Descrição — Uma descrição do que a métrica está medindo.
- Atributos — O conjunto de atributos (dimensões) emitidos com a métrica.

Registro em log

O AWS SDK para Kotlin configura um registrador compatível com [SLF4J](#) como o padrão do provedor `LoggerProvider` de telemetria. Com SLF4J, que é uma camada de abstração, você pode usar qualquer um dos vários sistemas de registro em tempo de execução. [Os sistemas de registro suportados incluem Java Logging APIs, Log4j 2 e Logback.](#)

Warning

Recomendamos que você use o registro de conexões somente para fins de depuração. (O registro de cabos é discutido abaixo.) Desative-o em seus ambientes de produção porque ele pode registrar dados confidenciais, como endereços de e-mail, tokens de segurança, chaves de API, senhas e AWS Secrets Manager segredos. O registro de conexões registra a solicitação ou resposta completa sem criptografia, mesmo para uma chamada HTTPS.

Para solicitações ou respostas grandes (como o upload de um arquivo para o Amazon S3), o registro detalhado de conexões também pode afetar significativamente o desempenho do seu aplicativo.

Exemplo de configuração de registro do Log4j 2

Embora qualquer biblioteca SLF4J de registros compatível possa ser usada, este exemplo permite a saída de log do SDK em programas JVM usando o Log4j 2:

Dependências do Gradle

(Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.)

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

Arquivo de configuração do Log4j 2

Crie um arquivo nomeado `log4j2.xml` em seu `resources` diretório (por exemplo, `<project-dir>/src/main/resources`). Adicione a seguinte configuração XML ao arquivo:

```
<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>
```

Essa configuração inclui o `%X` especificador no `pattern` atributo que ativa o registro do MDC (contexto de diagnóstico mapeado).

O SDK adiciona os seguintes elementos do MDC para cada operação.

rpc

O nome da RPC invocada, por exemplo. `S3.GetObject`

sdkInvocationId

Uma ID exclusiva atribuída pelo cliente do serviço para a operação. O ID correlaciona todos os eventos de registro relacionados à invocação de uma única operação.

Especifique o modo de registro para mensagens em nível de fio

Por padrão, o AWS SDK para Kotlin não registra mensagens em nível de fio porque elas podem conter dados confidenciais de solicitações e respostas da API. No entanto, às vezes você precisa desse nível de detalhe para fins de depuração.

Com o Kotlin SDK, você pode definir um modo de log no código ou usar as configurações do ambiente para habilitar mensagens de depuração para o seguinte:

- Solicitações HTTP
- Respostas HTTP

O modo de log é apoiado por um campo de bits em que cada bit é um sinalizador (modo) e os valores são aditivos. Você pode combinar um modo de solicitação e um modo de resposta.

Definir o modo de registro no código

Para optar pelo registro adicional, defina a `logMode` propriedade ao criar um cliente de serviço.

O exemplo a seguir mostra como habilitar o registro de solicitações (com o corpo) e a resposta (sem o corpo).

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

// ...

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

Um valor de modo de log definido durante a construção do cliente de serviço substitui qualquer valor de modo de log definido do ambiente.

Defina o modo de registro a partir do ambiente

Para definir um modo de log globalmente para todos os clientes de serviço não configurados explicitamente no código, use um dos seguintes:

- Propriedade do sistema JVM: `sdk.logMode`
- Variável de ambiente: `SDK_LOG_MODE`

Os seguintes valores que não diferenciam maiúsculas de minúsculas estão disponíveis:

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

Para criar um modo de log combinado usando as configurações do ambiente, você separa os valores com um símbolo pipe (`|`).

Por exemplo, os exemplos a seguir definem o mesmo modo de log do exemplo anterior.

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

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

Note

Você também deve configurar um registrador SLF4 J compatível e definir o nível de registro como `DEBUG` para ativar o registro em nível de fio.

Provedores de telemetria

Atualmente, o SDK oferece suporte a [OpenTelemetry](#)(OTel) como provedor. O SDK pode oferecer provedores de telemetria adicionais no futuro.

Tópicos

- [Configurar o provedor OpenTelemetry de telemetria baseado](#)

Configurar o provedor OpenTelemetry de telemetria baseado

O SDK para Kotlin fornece uma implementação da TelemetryProvider interface apoiada por OpenTelemetry

Pré-requisitos

Atualize as dependências do seu projeto para adicionar o OpenTelemetry provedor, conforme mostrado no seguinte trecho do Gradle. Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.

```
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")
}
```

Configurar o SDK

O código a seguir configura um cliente de serviço usando o provedor de OpenTelemetry telemetria.

```
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

Uma discussão sobre como configurar o OpenTelemetry SDK está fora do escopo deste guia. A [documentação OpenTelemetry Java](#) contém informações de configuração sobre as várias abordagens: [manual](#), automaticamente por meio do [agente Java](#) ou do [coletor](#) (opcional).

Recursos

Os recursos a seguir estão disponíveis para ajudar você a começar OpenTelemetry.

- [AWS Distro para OpenTelemetry - Página](#) inicial da AWS OTe L Distro
- [aws-otel-java-instrumentation](#)- AWS Distro para biblioteca de instrumentação OpenTelemetry Java
- [aws-otel-lambda](#)- camadas OpenTelemetry Lambda AWS gerenciadas
- [aws-otel-collector](#)- AWS Distro para OpenTelemetry colecionador
- [AWS Melhores práticas de observabilidade](#) - Melhores práticas gerais para observabilidade específica para AWS

Substituir a configuração do cliente de serviço

Depois que um [cliente de serviço é criado](#), o cliente de serviço usa uma configuração fixa para todas as operações. No entanto, às vezes você pode precisar substituir a configuração para uma ou mais operações específicas.

Cada cliente de serviço tem uma `withConfig` extensão para que você possa modificar uma cópia da configuração existente. A `withConfig` extensão retorna um novo cliente de serviço com uma configuração modificada. O cliente original existe de forma independente e usa sua configuração original.

O exemplo a seguir mostra a criação de uma `S3Client` instância que chama duas operações.

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

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

O trecho a seguir mostra como substituir a configuração de uma única operação. `listObjectsV2`

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

As chamadas de operação do `s3` cliente usam a configuração original que foi especificada quando o cliente foi criado. Sua configuração inclui o [registro de solicitações](#) e `us-west-2` region para a região.

A `listObjectsV2` invocação no `overriddenS3` cliente usa as mesmas configurações do `s3` cliente original, exceto para a Região, que agora é. `eu-central-1`

Ciclo de vida de um cliente sobrecarregado

No exemplo anterior, o `s3` cliente e o `overriddenS3` cliente são independentes um do outro. As operações podem ser invocadas em qualquer cliente enquanto permanecerem abertas. Cada um usa uma configuração separada, mas pode compartilhar recursos subjacentes (como um mecanismo HTTP), a menos que eles também sejam substituídos.

Você fecha um cliente com uma configuração substituída e o cliente original separadamente. Você pode fechar um cliente com a configuração substituída antes ou depois de fechar o cliente original. A menos que você precise usar um cliente com configuração substituída por muito tempo, recomendamos que você envolva seu ciclo de vida com o método. `use` O `use` método garante que o cliente seja fechado caso ocorram exceções.

Recursos compartilhados entre clientes

Quando você cria um cliente de serviço usando `withConfig`, ele pode compartilhar recursos com o cliente original. Por outro lado, quando você cria um cliente usando [FromEnvironment ou o configura explicitamente, o cliente usa recursos](#) independentes. Recursos como mecanismos HTTP e provedores de credenciais são compartilhados, a menos que sejam substituídos no bloco `withConfig`.

Como o ciclo de vida de cada cliente é independente, os recursos compartilhados permanecem abertos e utilizáveis até que o último cliente seja fechado. Portanto, é importante fechar clientes de serviços substituídos quando não precisar mais deles. Isso evita que recursos compartilhados permaneçam abertos e consumam recursos do sistema, como memória, conexão e ciclos de CPU.

O exemplo a seguir mostra recursos compartilhados e independentes.

Os `overriddenS3` clientes `s3` e compartilham a mesma instância do provedor de credenciais, incluindo sua configuração de cache. Chamadas feitas por meio de credenciais de `overriddenS3` reutilizam se o valor em cache ainda estiver atualizado a partir das chamadas feitas pelo cliente. `s3`

O mecanismo HTTP não é compartilhado entre os dois clientes. Cada cliente tem um mecanismo HTTP independente porque ele foi substituído na `withConfig` chamada.

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

s3.listBuckets { ... }

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


Usar a SDK

Esta seção fornece as informações básicas necessárias para usar AWS SDK para Kotlin o.

Tópicos

- [Fazer solicitações.](#)
- [Corrotinas](#)
- [Operações de streaming](#)
- [Paginação](#)
- [Waiters](#)
- [Tratamento de erros](#)
- [Solicitações pré-assinadas](#)
- [Solução de problemas do FAQs](#)

Fazer solicitações.

Use um cliente de serviço para fazer solicitações a um AWS service (Serviço da AWS). O AWS SDK para Kotlin fornece idiomas específicos de domínio (DSLs) seguindo um padrão de [construtor de tipos seguros](#) para criar solicitações. Estruturas aninhadas de solicitações também podem ser acessadas por meio de seus DSLs

O exemplo a seguir mostra como criar uma entrada de operação do 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 Sobrecargas na interface de serviço

Cada operação sem streaming na interface do cliente de serviço tem uma DSL sobrecarga para que você não precise criar uma solicitação separada.

Exemplo de criação de um bucket do Amazon Simple Storage Service (Amazon S3) com a função sobrecarregada:

```

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

```

Isso é equivalente a:

```

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

s3client.createBucket(request)

```

Solicitações sem entradas necessárias

As operações que não têm entradas obrigatórias podem ser chamadas sem a necessidade de passar por um objeto de solicitação. Isso geralmente é possível com operações do tipo lista, como a operação do Amazon `listBuckets` API S3.

Por exemplo, as três declarações a seguir são equivalentes:

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

Corrotinas

O AWS SDK para Kotlin é assíncrono por padrão. O SDK for Kotlin usa `suspend` funções para todas as operações, que devem ser chamadas a partir de uma corrotina.

Para obter um guia mais detalhado sobre corrotinas, consulte a documentação [oficial](#) do Kotlin.

Fazendo solicitações simultâneas

O construtor de corrotinas [assíncronas](#) pode ser usado para iniciar solicitações simultâneas nas quais você se preocupa com os resultados. `async` retorna um [Deferred](#), que representa um future leve e sem bloqueios que representa a promessa de fornecer um resultado posteriormente.

Se você não se importa com os resultados (apenas com a conclusão de uma operação), você pode usar o construtor de corrotinas de [inicialização](#). `launch` é conceitualmente semelhante a `async`. A diferença é que o `launch` retorna um [Job](#) e não carrega nenhum valor resultante, enquanto `async` retorna `Deferred`.

Veja a seguir um exemplo de como fazer solicitações simultâneas ao Amazon S3 usando [headObject](#) operação para obter o tamanho do conteúdo de duas chaves:

```
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")
}
```

Fazendo solicitações de bloqueio

Para fazer chamadas de serviço a partir de um código existente que não usa corrotinas e implementa um modelo de encadeamento diferente, você pode usar o construtor de corrotinas. [runBlocking](#) Um exemplo de um modelo de segmentação diferente é usar a abordagem tradicional de executores/

futuros do Java. Talvez seja necessário usar essa abordagem se estiver combinando código ou bibliotecas Java e Kotlin.

Como o próprio nome sugere, esse `runBlocking` construtor lança uma nova corrotina e bloqueia o encadeamento atual até que ele seja concluído.

Warning

`runBlocking` geralmente não deve ser usado a partir de uma corrotina. Ele foi projetado para conectar o código de bloqueio regular às bibliotecas escritas em estilo suspenso (como nas funções e testes principais).

Operações de streaming

No AWS SDK para Kotlin, os dados binários (fluxos) são representados como um [ByteStream](#) tipo, que é um fluxo abstrato de bytes somente para leitura.

Respostas de streaming

As respostas com um stream binário (como a operação do Amazon Simple Storage Service (Amazon [GetObject](#) API S3)) são tratadas de forma diferente de outros métodos. Esses métodos usam uma função lambda que manipula a resposta em vez de retorná-la diretamente. Isso limita o escopo da resposta à função e simplifica o gerenciamento da vida útil do chamador e do SDK tempo de execução.

Depois que a função lambda retorna, todos os recursos, como a HTTP conexão subjacente, são liberados. (Eles não `ByteStream` devem ser acessados após o retorno do lambda e não devem ser retirados do fechamento.) O resultado da chamada é o que o lambda retorna.

O exemplo de código a seguir mostra a [getObject](#) função recebendo um parâmetro lambda, que manipula a resposta.

```
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")
```

O `ByteStream` tipo tem as seguintes extensões para formas comuns de consumi-lo:

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

Tudo isso está definido no `aws.smithy.kotlin.runtime.content` pacote.

Solicitações de streaming

Para fornecer um `ByteStream`, também existem vários métodos de conveniência, incluindo os seguintes:

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

Tudo isso está definido no `aws.smithy.kotlin.runtime.content` pacote.

O exemplo de código a seguir mostra o uso de métodos de `ByteStream` conveniência que fornecem a propriedade `body` na criação de um [PutObjectRequest](#):

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

Paginação

Muitas AWS operações retornam resultados paginados quando a carga é muito grande para ser retornada em uma única resposta. AWS SDK para Kotlin inclui [extensões](#) para a interface do cliente de serviço que paginam automaticamente os resultados para você. Você só precisa escrever o código que processa os resultados.

A paginação é exposta como um [fluxo](#) `<T>` para que você possa aproveitar as transformações idiomáticas do Kotlin para coleções assíncronas (como, `e`), `map`, `filter`, `take`. As exceções são transparentes, o que faz com que o tratamento de erros pareça uma API chamada normal, e o cancelamento segue o cancelamento cooperativo geral de corrotinas. Para obter mais informações, consulte [fluxos](#) e [exceções de fluxo](#) no guia oficial.

Note

Os exemplos a seguir usam o Amazon S3. No entanto, os conceitos são os mesmos para qualquer serviço que tenha um ou mais APIs paginados. Todas as extensões de paginação são definidas no `aws.sdk.kotlin.<service>.paginators` pacote (como `aws.sdk.kotlin.dynamodb.paginators`).

O exemplo de código a seguir mostra como você pode processar a resposta paginada da chamada da função [listObjectsV2Paginated](#).

Importações

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

Código

```
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}")
    }
```

Waiters

Os garçons são uma abstração do lado do cliente usada para pesquisar um recurso até que o estado desejado seja alcançado ou até que seja determinado que o recurso não entrará no estado desejado. Essa é uma tarefa comum quando se trabalha com serviços que acabam sendo consistentes, como o Amazon Simple Storage Service (Amazon S3), ou serviços que criam recursos de forma assíncrona, como o Amazon. EC2

Escrever uma lógica para pesquisar continuamente o status de um recurso pode ser complicado e propenso a erros. O objetivo dos garçons é transferir essa responsabilidade do código do cliente para o AWS SDK para Kotlin, que tem um conhecimento profundo dos aspectos de cronometragem da operação. AWS

Note

Os exemplos a seguir usam o Amazon S3. No entanto, os conceitos são os mesmos para qualquer um AWS service (Serviço da AWS) que tenha um ou mais garçons definidos. Todas as extensões são definidas no `aws.sdk.kotlin.<service>.waiters` pacote (como `aws.sdk.kotlin.dynamodb.waiters`). Eles também seguem uma convenção de nomenclatura padrão (`waitUntil<Condition>`).

O exemplo de código a seguir mostra o uso de uma função de espera que permite evitar escrever a lógica de pesquisa.

Importações


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

Código

```
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

Cada método de espera retorna uma `Outcome` instância que pode ser usada para obter a resposta final que corresponde ao alcance da condição desejada. Um resultado também contém detalhes adicionais, como o número de tentativas feitas para alcançar o estado desejado.

Tratamento de erros

Entender como e quando AWS SDK para Kotlin as exceções são lançadas é importante para criar aplicativos de alta qualidade usando o SDK. As seções a seguir descrevem os diferentes casos de exceções lançados pelo SDK e como lidar com eles de forma adequada.

Exceções de serviço

A exceção mais comum é `AwsServiceException`, da qual todas as exceções específicas do serviço (como `S3Exception`) são herdadas. Essa exceção representa uma resposta de erro de um AWS service (Serviço da AWS). Por exemplo, se você tentar encerrar uma EC2 instância da Amazon que não existe, a Amazon EC2 retornará uma resposta de erro. Os detalhes da resposta ao erro estão incluídos no `AwsServiceException` que foi lançado.

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

Exceções do cliente

`ClientException` indica que ocorreu um problema dentro do código do AWS SDK para Kotlin cliente, ao tentar enviar uma solicitação para AWS ou ao tentar analisar uma resposta do AWS. `ClientException` geralmente é mais grave do que a `AwsServiceException` e indica que um grande problema está impedindo o cliente de processar as chamadas de serviço para Serviços da AWS. Por exemplo, o AWS SDK para Kotlin lança um `ClientException` se não conseguir analisar uma resposta de um serviço.

Metadados de erro

Cada exceção de serviço e exceção de cliente tem a `sdkErrorMetadata` propriedade. Esse é um pacote de propriedades digitado que pode ser usado para recuperar detalhes adicionais sobre o erro.

Existem várias extensões predefinidas diretamente para o `AwsErrorMetadata` tipo, incluindo, mas não se limitando às seguintes:

- `sdkErrorMetadata.requestId`— o ID de solicitação exclusivo
- `sdkErrorMetadata.errorMessage`— a mensagem legível por humanos (geralmente corresponde a `Exception.message`, mas pode conter mais informações se a exceção for desconhecida pelo serviço)
- `sdkErrorMetadata.protocolResponse`— A resposta bruta do protocolo

O exemplo a seguir demonstra o acesso aos metadados de erro.

```
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}")
}
```

Solicitações pré-assinadas

Você pode pré-assinar solicitações para algumas AWS API operações para que outro chamador possa usar a solicitação posteriormente sem apresentar suas próprias credenciais.

Por exemplo, suponha que Alice tenha acesso a um objeto do Amazon Simple Storage Service (Amazon S3) e queira compartilhar temporariamente o acesso ao objeto com Bob. Alice pode gerar uma `GetObject` solicitação pré-assinada para compartilhar com Bob para que ele possa baixar o objeto sem precisar acessar as credenciais de Alice.

Princípios básicos da pré-assinatura

O SDK for Kotlin fornece métodos de extensão em clientes de serviço para pré-assinar solicitações. Todas as solicitações pré-assinadas exigem uma duração que represente por quanto tempo a solicitação assinada é válida. Após o término da duração, a solicitação pré-assinada expira e gera um erro de autenticação se executada.

O código a seguir mostra um exemplo que cria uma `GetObject` solicitação pré-assinada para o Amazon S3. A solicitação é válida por 24 horas após a criação.

```
val s3 = S3Client.fromEnvironment()

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

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

O método `presignGetObject` de extensão retorna um `HttpRequest` objeto. O objeto de solicitação contém o preassinado URL em que a operação pode ser invocada. Outro chamador pode usar a URL (ou a solicitação inteira) em uma base de código ou ambiente de linguagem de programação diferente.

Depois de criar a solicitação pré-assinada, use um HTTP cliente para invocar uma solicitação. A API invocação de uma HTTP GET solicitação depende do HTTP cliente. O exemplo a seguir usa o método Kotlin `URL.readText`.

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

```
println(objectContents)
```

Configuração avançada de pré-assinatura

No SDK, cada método que pode pré-assinar solicitações tem uma sobrecarga que você pode usar para fornecer opções avançadas de configuração, como uma implementação específica do signatário ou parâmetros de assinatura detalhados.

O exemplo a seguir mostra uma `GetObject` solicitação do Amazon S3 que usa a variante CRT signer e especifica uma data de assinatura futura.

```
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
}
```

A solicitação pré-assinada devolvida tem data de 24 horas e não é válida antes disso. Ele expira 8 horas depois disso.

Pré-assinatura POST e solicitações PUT

Muitas operações presignáveis exigem apenas a URL e devem ser executadas como HTTP GET solicitações. Algumas operações, no entanto, usam um corpo e, em alguns casos, devem ser executadas como uma HTTP PUT solicitação HTTP POST ou junto com cabeçalhos. A pré-assinatura dessas solicitações é idêntica às GET solicitações pré-assinadas, mas invocar a solicitação pré-assinada é mais complicado.

Aqui está um exemplo de pré-assinatura de uma solicitação do S3: `PutObject`

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = PutObjectRequest {
    bucket = "foo"
```

```

    key = "bar"
}

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

```

O retornado `HttpRequest` tem um valor de método de `HttpMethod.PUT` e inclui cabeçalhos a URL e que devem ser incluídos na invocação futura da solicitação. HTTP Você pode passar essa solicitação para um chamador que pode executá-la em uma base de código ou ambiente de linguagem de programação diferente.

Depois de criar o pré-assinado POST ou a PUT solicitação, use um HTTP cliente para invocar uma solicitação. A API invocação de uma PUT solicitação POST or URL depende do HTTP cliente usado. O exemplo a seguir usa um [OkHttp HTTPcliente](#) e inclui um corpo que contém `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()

```

Operações que SDK podem preassinar

Atualmente, o SDK for Kotlin suporta a pré-assinatura das seguintes API operações que precisam ser chamadas com o método associado. HTTP

AWS service (Serviço da AWS)	Operação	SDK método de extensão	HTTP Método de uso
Amazon S3	GetObject	presignGetObject	HTTP GET
Amazon S3	PutObject	presignPutObject	HTTP PUT
Amazon S3	UploadPart	presignUploadPart	HTTP PUT

AWS service (Serviço da AWS)	Operação	SDK método de extensão	HTTP Método de uso
AWS Security Token Service	GetCallerIdentity	presignGetCallerId entidade	HTTP POST
Amazon Polly	SynthesizeSpeech	presignSynthesizeSpeech	HTTP POST

Solução de problemas do FAQs

Ao usar o AWS SDK para Kotlin em seus aplicativos, você pode encontrar alguns dos problemas listados neste tópico. Use as sugestões a seguir para ajudar a descobrir a causa raiz e resolver o erro.

Como faço para corrigir problemas de “conexão fechada”?

Você pode encontrar problemas de “conexão fechada” como exceções, como um dos seguintes tipos:

- `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)`

Essas exceções indicam que uma TCP conexão do SDK com um serviço foi fechada ou redefinida inesperadamente. A conexão pode ter sido fechada pelo seu host, pelo AWS serviço ou por uma parte intermediária, como um NAT gateway, proxy ou balanceador de carga.

Esses tipos de exceções são repetidos automaticamente, mas ainda podem aparecer nos SDK registros, dependendo da sua configuração de registro. Se a exceção for inserida em seu código, isso indica que a estratégia de repetição ativa esgotou seus limites configurados, como o máximo de tentativas ou o repositório de tokens de repetição. Consulte a [the section called “Repetições”](#) seção deste guia para obter mais informações sobre estratégias de repetição. Veja também [the section called “Por que as exceções são lançadas antes de atingir o máximo de tentativas?”](#) o tópico?.

Por que as exceções são lançadas antes de atingir o máximo de tentativas?

Às vezes, você pode ver exceções que esperava que fossem repetidas, mas que, em vez disso, foram descartadas. Nessas situações, as etapas a seguir podem ajudar a resolver o problema.

- Verifique se a exceção pode ser repetida. Algumas exceções não podem ser repetidas, como aquelas que indicam uma solicitação de serviço malformada, falta de permissões e recursos inexistentes, como exemplos. O SDK não repete automaticamente esses tipos de exceções. Se você estiver capturando uma exceção herdada de `SdkBaseException`, você pode verificar a propriedade booleana `SdkBaseException.sdkErrorMetadata.isRetryable` para verificar se ela SDK determinou que a exceção pode ser repetida.
- Verifique se a exceção está sendo inserida em seu código. Algumas exceções aparecem nas mensagens de log como informações, mas na verdade não são incluídas em seu código. Por exemplo, exceções que podem ser repetidas, como erros de limitação, podem ser registradas, pois funcionam automaticamente em vários ciclos. SDK backoff-and-retry A invocação de uma SDK operação gera uma exceção somente se não for tratada pelas configurações de repetição definidas.
- Verifique suas configurações de repetição definidas. Consulte a [the section called “Repetições”](#) seção deste guia para obter mais informações sobre estratégias e políticas de repetição. Certifique-se de que seu código esteja usando as configurações esperadas ou os padrões automáticos.
- Considere ajustar suas configurações de nova tentativa. Depois de verificar os itens anteriores, mas o problema não ter sido resolvido, você pode considerar ajustar as configurações de nova tentativa.
 - Aumente o número máximo de tentativas. Por padrão, o número máximo de tentativas de uma operação é 3. Se você achar que isso não é suficiente e as exceções ainda estão ocorrendo na configuração padrão, considere aumentar a `retryStrategy.maxAttempts` propriedade na configuração do seu cliente. Consulte [the section called “Máximo de tentativas”](#) para obter mais informações.
 - Aumente as configurações de atraso. Algumas exceções podem ser repetidas muito rapidamente antes que a condição subjacente tenha a chance de ser resolvida. Se você suspeitar que esse seja o caso, considere aumentar as `retryStrategy.delayProvider.maxBackoff` propriedades `retryStrategy.delayProvider.initialDelay` ou na configuração do seu cliente. Consulte [the section called “Atrasos e recuos”](#) para obter mais informações.

- Desative o modo disjuntor. Por padrão, SDK mantém um conjunto de tokens para cada cliente de serviço. Quando ele SDK tenta uma solicitação e ela falha com uma exceção que pode ser repetida, a contagem de tokens é diminuída; quando a solicitação é bem-sucedida, a contagem de tokens é incrementada.

Por padrão, se esse token bucket atingir 0 tokens restantes, o circuito será interrompido. Depois que o circuito é interrompido, as novas tentativas SDK de desativação e todas as solicitações atuais e subseqüentes que falharem na primeira tentativa geram imediatamente uma exceção. As SDK novas tentativas reativadas após tentativas iniciais bem-sucedidas retornam capacidade suficiente para o token bucket. Esse comportamento é intencional e foi projetado para evitar novas tentativas durante interrupções e recuperação do serviço.

Se você preferir SDK continuar tentando novamente até o máximo de tentativas configuradas, considere desativar o modo disjuntor definindo a `retryStrategy.tokenBucket.useCircuitBreakerMode` propriedade como `false` na configuração do cliente. Com essa propriedade definida como `false`, o SDK cliente espera até que o token bucket atinja a capacidade suficiente, em vez de abandonar novas tentativas, o que pode levar a uma exceção quando houver 0 tokens restantes.

Como faço para corrigir `NoSuchMethodError` ou `NoClassDefFoundError`?

O SDK depende de várias dependências AWS e de terceiros para funcionar corretamente. Se as dependências esperadas não estiverem presentes no tempo de execução ou forem uma versão inesperada, você poderá ver a exceção do `NoSuchMethodError` tempo de execução.

Os conflitos de dependência geralmente se enquadram em duas categorias: SDK /Conflitos de dependência de Smithy e conflitos de dependência de terceiros.

Quando você cria um aplicativo Kotlin, normalmente gerencia dependências com o Gradle. Adicionar uma dependência de um cliente de SDK serviço ao seu aplicativo deve resolver e incluir automaticamente todas as dependências transitivas. Se seu aplicativo tiver outras dependências, elas poderão entrar em conflito com as dependências necessárias para o SDK (por exemplo, `OkHttp` é um HTTP cliente comumente usado do qual SDK depende).

Para resolver esses problemas, talvez seja necessário resolver explicitamente uma versão específica da dependência ou dependências ocultas em um namespace local para evitar o conflito. A resolução de dependências do Gradle é um tópico complexo que é discutido nas seções a seguir do Manual do usuário do Gradle.

- [Entendendo a resolução de dependências](#)
- [Restrições de dependência e resolução de conflitos](#)
- [Alinhando versões de dependência](#)

SDK/Conflitos de dependência de Smithy

Em geral, os módulos dependem SDK de outros SDK módulos com o mesmo número de versão. Por exemplo, `aws.sdk.kotlin:s3:1.2.3` depende de `aws.sdk.kotlin:aws-http:1.2.3`, o que depende de `aws.sdk.kotlin:aws-core:1.2.3` e assim por diante.

Além disso, SDK os módulos também contam com versões específicas e unificadas do módulo Smithy. Esses números de versão do Smithy não estão sincronizados com os números de SDK versão, mas ainda devem corresponder à versão esperada pelo SDK. Por exemplo, `aws.sdk.kotlin:s3:1.2.3` pode depender de `aws.smithy.kotlin:serde:1.1.1`, o que depende de `aws.smithy.kotlin:runtime-core:1.1.1` e assim por diante.

Se algum desses números de versão for incompatível, você poderá encontrar conflitos de dependência. Certifique-se de atualizar todas as suas SDK dependências em uníssono e também atualizar todas as dependências explícitas do Smithy em uníssono. Considere usar nosso [catálogo de versões do Gradle](#) para manter as versões sincronizadas e eliminar suposições no mapeamento entre as versões SDK e do Smithy. Consulte o [the section called “Crie arquivos de construção do projeto”](#) tópico para obter mais informações e exemplos.

Além disso, esteja ciente de que pequenas alterações de versão nos módulos SDK /Smithy podem conter alterações significativas, conforme descrito na política de controle de versão [do SDK](#). Ao atualizar entre versões secundárias, tome cuidado extra para examinar os registros de alterações e validar minuciosamente o comportamento do tempo de execução.

Eu vejo um **NoClassDefFoundError** para **okhttp3/coroutines/ExecuteAsyncKt**

Se você ver esse erro, provavelmente significa que você não configurou seu cliente de serviço para usar `OkHttp4Engine` o. [Leia a documentação](#) sobre como configurar o Gradle e usá-lo `OkHttp4Engine` em seu código.

Trabalhe com Serviços da AWS o uso do AWS SDK para Kotlin

Este capítulo contém informações sobre como trabalhar Serviços da AWS usando o SDK for Kotlin.

Sumário

- [Trabalhe com o Amazon S3 usando o AWS SDK para Kotlin](#)
- [Proteção da integridade de dados com somas de verificação](#)
 - [Fazer upload de um objeto](#)
 - [Usar um valor de soma de verificação pré-calculado](#)
 - [Carregamentos fracionados](#)
 - [Fazer download de um objeto](#)
 - [Validação assíncrona](#)
- [Trabalhe com pontos de acesso multirregionais do Amazon S3 usando o for Kotlin SDK](#)
 - [Trabalhe com pontos de acesso multirregionais](#)
 - [Trabalhe com objetos e pontos de acesso multirregionais](#)
- [Trabalhe com o DynamoDB usando o AWS SDK para Kotlin](#)
- [Use AWS endpoints baseados em conta](#)
- [Mapeie classes para itens do DynamoDB usando o DynamoDB Mapper \(Developer Preview\)](#)
 - [Comece a usar o DynamoDB Mapper](#)
 - [Adicionar dependências](#)
 - [Crie e use um mapeador](#)
 - [Defina um esquema com anotações de classe](#)
 - [Invocar operações](#)
 - [Trabalhe com respostas paginadas](#)
 - [Configurar o DynamoDB Mapper](#)
 - [Use interceptores](#)
 - [Entenda o pipeline de solicitações](#)
 - [Hooks](#)
 - [Ganchos somente para leitura](#)

- [Modificar ganchos](#)
- [Ordem de execução](#)
- [Exemplo de configuração](#)
- [Gere um esquema a partir de anotações](#)
 - [Aplique o plugin](#)
 - [Configurar o plug-in](#)
 - [Anote as aulas](#)
 - [Anotações de classe](#)
 - [Anotações de propriedade](#)
 - [Definir um conversor de itens personalizado](#)
- [Defina esquemas manualmente](#)
 - [Definir um esquema no código](#)
- [Use índices secundários com o DynamoDB Mapper](#)
 - [Definir um esquema para um índice secundário](#)
 - [Use índices secundários nas operações](#)
- [Use expressões](#)
 - [Use expressões em operações](#)
 - [Componentes do DSL](#)
 - [Atributos](#)
 - [Igualdades e desigualdades](#)
 - [Intervalos e conjuntos](#)
 - [Lógica booleana](#)
 - [Funções e propriedades](#)
 - [Filtros de chave de classificação](#)

Trabalhe com o Amazon S3 usando o AWS SDK para Kotlin

[Sua interface principal para o Amazon Simple Storage Service para o Kotlin SDK é o S3Client](#). Use os `S3Client` mesmos outros clientes de serviço no SDK para fazer [solicitações](#) ao Amazon S3.

- a SDK [APIreferência Kotlin para S3](#).
- o [Guia do usuário do serviço S3](#) e a [APIreferência do serviço](#).

Os tópicos a seguir apresentam exemplos de códigos guiados para Kotlin selecionados SDK APIs que funcionam com o S3.

Tópicos

- [Proteção da integridade de dados com somas de verificação](#)
- [Trabalhe com pontos de acesso multirregionais do Amazon S3 usando o for Kotlin SDK](#)

Proteção da integridade de dados com somas de verificação

O Amazon Simple Storage Service (Amazon S3) oferece a capacidade de especificar uma soma de verificação ao fazer upload de um objeto. Quando você especifica uma soma de verificação, ela é armazenada com o objeto e pode ser validada quando o objeto é baixado.

As somas de verificação fornecem uma camada adicional de integridade de dados quando você transfere arquivos. Com somas de verificação, você pode verificar a consistência de dados confirmando que o arquivo recebido corresponde ao arquivo original. [Para obter mais informações sobre somas de verificação com o Amazon S3, consulte o Guia do usuário do Amazon Simple Storage Service, incluindo os algoritmos compatíveis.](#)

Você tem a flexibilidade de escolher o algoritmo mais adequado às suas necessidades e deixar que o SDK calcule a soma de verificação. Como alternativa, você pode fornecer um valor de soma de verificação pré-computado usando um dos algoritmos compatíveis.

Note

A partir da versão 1.4.0 do AWS SDK para Kotlin, o SDK fornece proteções de integridade padrão calculando automaticamente uma CRC32 soma de verificação para uploads. O SDK calcula essa soma de verificação se você não fornecer um valor de soma de verificação pré-calculado ou se não especificar um algoritmo que o SDK deva usar para calcular uma soma de verificação.

O SDK também fornece configurações globais para proteções de integridade de dados que você pode definir externamente, sobre as quais você pode ler no Guia de referência de [ferramentas AWS SDKs e ferramentas](#).

Discutimos somas de verificação em duas fases de solicitação: upload de um objeto e download de um objeto.

Fazer upload de um objeto

Você carrega objetos para o Amazon S3 com o SDK para Kotlin usando a função [putObject](#) com um parâmetro de solicitação. O tipo de dados da solicitação fornece a propriedade `checksumAlgorithm` para habilitar o cálculo da soma de verificação.

O trecho de código a seguir mostra uma solicitação para carregar um objeto com uma CRC32 soma de verificação. Quando o SDK envia a solicitação, ele calcula a CRC32 soma de verificação e carrega o objeto. O Amazon S3 armazena a soma de verificação com o objeto.

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

Se você não fornecer um algoritmo de soma de verificação com a solicitação, o comportamento da soma de verificação varia de acordo com a versão do SDK que você usa, conforme mostrado na tabela a seguir.

Comportamento da soma de verificação quando nenhum algoritmo de soma de verificação é fornecido

Versão Kotlin SDK	Comportamento da soma de verificação
anterior a 1.4.0	O SDK não calcula automaticamente uma soma de verificação baseada em CRC e a fornece na solicitação.
1.4.0 ou posterior	O SDK usa o CRC32 algoritmo para calcular a soma de verificação e a fornece na solicitação. O Amazon S3 valida a integridade da transferência computando sua própria soma de CRC32 verificação e a compara com a soma de verificação fornecida pelo SDK. Se as somas

Versão Kotlin SDK	Comportamento da soma de verificação
	de verificação corresponderem, a soma de verificação será salva com o objeto.

Usar um valor de soma de verificação pré-calculado

Um valor de soma de verificação pré-calculado fornecido com a solicitação desabilita a computação automática pelo SDK e, em vez disso, usa o valor fornecido.

O exemplo a seguir mostra uma solicitação com uma soma de SHA256 verificação pré-calculada.

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

Se o Amazon S3 determinar que o valor da soma de verificação está incorreto para o algoritmo especificado, o serviço retornará uma resposta de erro.

Carregamentos fracionados

Você também pode usar somas de verificação com carregamentos fracionados.

Você deve especificar o algoritmo de soma de verificação na `CreateMultipartUpload` solicitação e em cada `UploadPart` solicitação. Como etapa final, você deve especificar a soma de verificação de cada parte no `CompleteMultipartUpload`. O exemplo a seguir mostra como criar um carregamento fracionado com o algoritmo de soma de verificação especificado.

```
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 = ByteString.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
    }
}
```

Fazer download de um objeto

Quando você usa o método [getObject](#) para baixar um objeto, o SDK valida automaticamente a soma de verificação o valor da chave é. `ChecksumMode enabled` quando a `checksumMode` propriedade do construtor do `GetObjectRequest` está definida como `ChecksumMode.Enabled`.

A solicitação no trecho a seguir direciona o SDK a validar a soma de verificação na resposta calculando a soma de verificação e comparando os valores.

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

Se o objeto não tiver sido carregado com uma soma de verificação, nenhuma validação ocorrerá.

Se você usa uma versão 1.4.0 ou posterior do SDK, o SDK verifica automaticamente a integridade das `getObject` solicitações sem adicioná-las `checksumMode = ChecksumMode.Enabled` à solicitação.

Validação assíncrona

Como o SDK para Kotlin usa respostas de streaming ao baixar um objeto do Amazon S3, a soma de verificação será calculada à medida que você consome o objeto. Portanto, você deve consumir o objeto para que a soma de verificação seja validada.

O exemplo a seguir mostra como validar uma soma de verificação consumindo totalmente a resposta.

```
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.
}
```

Por outro lado, o código no exemplo a seguir não usa o objeto de forma alguma, então a soma de verificação não é validada.

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

Se a soma de verificação calculada pelo SDK não corresponder à soma de verificação esperada enviada com a resposta, o SDK lançará uma `ChecksumMismatchException`.

Trabalhe com pontos de acesso multirregionais do Amazon S3 usando o for Kotlin SDK

Os pontos de acesso multirregionais do Amazon S3 fornecem um endpoint global que as aplicações podem usar para atender a solicitações de buckets do S3 localizados em várias Regiões da AWS. Você pode usar pontos de acesso multirregionais para criar aplicações de várias regiões com

a mesma arquitetura usada em uma única região e, em seguida, executar essas aplicações em qualquer lugar do mundo.

O Guia do usuário do Amazon S3 contém mais informações básicas sobre pontos de acesso [multirregionais](#).

Trabalhe com pontos de acesso multirregionais

Para criar um ponto de acesso multirregional, comece especificando um bucket em cada AWS região que você deseja atender às solicitações. O trecho a seguir cria dois buckets.

Criar buckets

A função a seguir cria dois buckets para trabalhar com o ponto de acesso multirregional. Um bucket está na Região us-east-1 e o outro está na Região us-west-1.

A criação do cliente S3 que foi transmitido como primeiro argumento é mostrada no primeiro exemplo abaixo [the section called “Trabalhar com objetos do ”](#).

```
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.waitUntilBucketExists {
    bucket = bucketName2
}
println(" Bucket [$bucketName2] created.")
}
}

```

Você usa o [cliente de controle S3 SDK](#) do Kotlin para criar, excluir e obter informações sobre pontos de acesso multirregionais.

Adicione uma dependência no artefato de controle do S3, conforme mostrado no trecho a seguir. (Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.)

```

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

```

Configure o cliente de controle S3 para trabalhar Região da AWS us-west-2 conforme mostrado no código a seguir. Todas as operações do cliente de controle do S3 devem ter como alvo a us-west-2 região.

```

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

```

Use o cliente de controle S3 para criar um ponto de acesso multirregional especificando os nomes dos buckets (criados anteriormente), conforme mostrado no código a seguir.

```

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"
    }

```

Como a criação de um ponto de acesso multirregional é uma operação assíncrona, você usa o token que recebe da resposta imediata para verificar o status do processo de criação.

Depois que a verificação de status retornar uma mensagem de sucesso, você poderá usar a `GetMultiRegionAccessPoint` operação para obter o alias do ponto de acesso multirregional. O alias é o último componente do ARN, que você precisa para operações em nível de objeto.

Use o token para verificar o status

Use o `DescribeMultiRegionAccessPointOperation` para verificar o status da última operação. Depois que o `requestStatus` valor se tornar "SUCCEEDED", você poderá trabalhar com o ponto de acesso multirregional.

```
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)
    }
}
```

Trabalhe com objetos e pontos de acesso multirregionais

Você usa o [cliente S3](#) para trabalhar com objetos em pontos de acesso multirregionais. Muitas das operações que você usa em objetos em buckets podem ser usadas em pontos de acesso

multirregionais. Para obter mais informações e uma lista completa de operações, consulte [Compatibilidade de pontos de acesso multirregionais com operações do S3](#).

As operações com pontos de acesso multirregionais são assinadas com o algoritmo de assinatura assimétrica Sigv4 (SIGv4a). Support para Sigv4a no AWS SDK para Kotlin momento requer assinatura com o CRT signatário, que é uma dependência separada. Para configurar o suporte para SIGv4a, adicione as seguintes dependências ao seu projeto. (Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.)

```
...
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")
...
```

Depois de adicionar as dependências, configure o cliente S3 para usar o algoritmo de assinatura SigV4a, conforme mostrado no código a seguir.

```
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
}
```

Depois de configurar o cliente S3, as operações que o S3 suporta para pontos de acesso multirregionais funcionam da mesma forma. A única diferença é que o parâmetro do bucket deve ser o do ponto ARN de acesso multirregional. Você pode obtê-lo no console ARN do Amazon S3 ou programaticamente, conforme mostrado anteriormente na `createMrap` função que retorna um. ARN

O exemplo de código a seguir mostra o ARN usado em uma `GetObject` operação.

```
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
    }
}
```

Trabalhe com o DynamoDB usando o AWS SDK para Kotlin

Use AWS endpoints baseados em conta

O DynamoDB [AWS oferece endpoints baseados em contas](#) que podem melhorar o desempenho usando AWS seu ID de conta para simplificar o roteamento de solicitações.

Para aproveitar esse recurso, você precisa usar a versão 1.3.37 ou superior do AWS SDK para Kotlin. Você pode encontrar a versão mais recente da SDK lista no [repositório central do Maven](#). Depois que uma versão compatível do SDK está ativa, ela usa automaticamente os novos endpoints.

Se você quiser cancelar o roteamento baseado em conta, você tem quatro opções:


- Configure um cliente de serviço do DynamoDB com `AccountIdEndpointMode` o definido como `DISABLED`
- Defina uma variável de ambiente.
- Defina uma propriedade JVM do sistema.
- Atualize a AWS configuração do arquivo de configuração compartilhado.

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

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

O Guia de referência de ferramentas AWS SDKs e ferramentas fornece mais informações sobre as últimas [três opções de configuração](#).

Mapeie classes para itens do DynamoDB usando o DynamoDB Mapper (Developer Preview)

 O DynamoDB Mapper é uma versão prévia para desenvolvedores. O recurso não está completo e está sujeito a alterações.


[O DynamoDB Mapper é uma biblioteca de alto nível que oferece mecanismos para mapear classes Kotlin para tabelas e índices do DynamoDB, semelhantes ao DynamoDB Enhanced Client ou ao AWS SDK for Java Object Persistence Model. AWS SDK for .NET](#)

Você define esquemas que descrevem seu objeto de dados e como convertê-los em itens do DynamoDB. Depois de definir o esquema, o DynamoDB Mapper fornece uma interface intuitiva para usar seus objetos em operações de criação, leitura, atualização ou exclusão CRUD () em suas tabelas e índices.

Tópicos

- [Comece a usar o DynamoDB Mapper](#)
- [Configurar o DynamoDB Mapper](#)
- [Gere um esquema a partir de anotações](#)
- [Defina esquemas manualmente](#)
- [Use índices secundários com o DynamoDB Mapper](#)
- [Use expressões](#)

Comece a usar o DynamoDB Mapper

 O DynamoDB Mapper é uma versão prévia para desenvolvedores. O recurso não está completo e está sujeito a alterações.

O tutorial a seguir apresenta os componentes básicos do DynamoDB Mapper e mostra como usá-lo em seu código.

Adicionar dependências

Para começar a trabalhar com o DynamoDB Mapper em seu projeto Gradle, adicione um plug-in e duas dependências ao seu arquivo. `build.gradle.kts`

(Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.)

```
// 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")
}
```

<Version>*Substitua pela versão mais recente do SDK. Para encontrar a versão mais recente do SDK, verifique a [versão mais recente em GitHub](#).

Note

Algumas dessas dependências são opcionais se você planeja definir esquemas manualmente. Consulte [the section called “Defina esquemas manualmente”](#) para obter mais informações e o conjunto reduzido de dependências.

Crie e use um mapeador

O DynamoDB Mapper usa o cliente do DynamoDB para interagir com AWS SDK para Kotlin o DynamoDB. Você precisa fornecer uma instância totalmente configurada ao criar uma [DynamoDbClient](#) instância de mapeador, conforme mostrado no seguinte trecho de código:

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

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

Depois de criar a instância do mapeador, você pode usá-la para obter a instância da tabela, conforme mostrado a seguir:

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

O código anterior faz referência a uma tabela DynamoDB nomeada `cars` com um esquema definido por `CarSchema` (discutiremos os esquemas abaixo). Depois de criar uma instância de tabela, você pode realizar operações nela. O trecho de código a seguir mostra dois exemplos de operações na `cars` tabela:

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

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

O código anterior cria um novo item na `cars` tabela. O código cria uma `Car` instância em linha usando a `Car` classe, cuja definição é mostrada abaixo. Em seguida, o código consulta a `cars` tabela em busca de itens cuja chave de partição esteja `Peugeot` e os imprime. As operações são [descritas com mais detalhes abaixo](#).

Defina um esquema com anotações de classe

Para várias classes do Kotlin, o SDK pode gerar esquemas automaticamente no momento da compilação usando o plug-in do DynamoDB Mapper Schema Generator para Gradle. Quando você usa o gerador de esquemas, o SDK inspeciona suas classes para inferir o esquema, o que alivia parte do clichê envolvido na definição manual de esquemas. [Você pode personalizar o esquema gerado usando anotações e configurações adicionais.](#)

Para gerar um esquema a partir de anotações, primeiro anote suas classes com [@DynamoDbItem](#) e quaisquer chaves com e. [@DynamoDbPartitionKey](#) [@DynamoDbSortKey](#) O código a seguir mostra a classe anotadaCar:

```
// 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
)
```

Depois de construir, você pode consultar o gerado automaticamenteCarSchema. Você pode usar a referência no getTable método do mapeador para obter uma instância de tabela, conforme mostrado a seguir:

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

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

Como alternativa, você pode obter a instância da tabela aproveitando um método de extensão [DynamoDbMapper](#) que é gerado automaticamente no momento da criação. Ao usar essa abordagem, você não precisa se referir ao esquema pelo nome. Conforme mostrado a seguir, o método de getCarsTable extensão gerado automaticamente retorna uma referência à instância da tabela:

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

Consulte [the section called “Gere um esquema”](#) para obter mais detalhes e exemplos.

Invocar operações

O DynamoDB Mapper suporta um subconjunto das operações disponíveis nos SDKs. `DynamoDbClient` As operações do Mapper são nomeadas da mesma forma que suas contrapartes no cliente SDK. Muitos membros de solicitação/resposta do mapeador são iguais aos seus colegas clientes do SDK, embora alguns tenham sido renomeados, digitados novamente ou totalmente eliminados.

Você invoca uma operação em uma instância de tabela usando uma sintaxe DSL, conforme mostrado a seguir:

```
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)
```

Você também pode invocar uma operação usando um objeto de solicitação explícito:

```
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)
```

Os dois exemplos de código anteriores são equivalentes.

Trabalhe com respostas paginadas

Algumas operações, como `query` e `scan` podem retornar, coleções de dados que podem ser grandes demais para serem retornadas em uma única resposta. Para garantir que todos os objetos

sejam processados, o DynamoDB Mapper fornece métodos de paginação, que não chamam o DynamoDB imediatamente, mas retornam a do tipo de resposta da operação, como [Flow](#) mostrado a seguir: `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) }
}
```


Muitas vezes, um fluxo de objetos é mais útil para a lógica de negócios do que um fluxo de respostas contendo objetos. O mapeador fornece um método de extensão em respostas paginadas para acessar o fluxo de objetos. Por exemplo, o código a seguir retorna um `Flow<Car>` em vez de `umFlow<ScanResponse<Car>>`, conforme mostrado anteriormente:

```
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) }
```

Configurar o DynamoDB Mapper

 O DynamoDB Mapper é uma versão prévia para desenvolvedores. O recurso não está completo e está sujeito a alterações.

O DynamoDB Mapper oferece opções de configuração que você pode usar para personalizar o comportamento da biblioteca de acordo com seu aplicativo.

Use interceptores

A biblioteca do DynamoDB Mapper define ganchos que você pode usar em estágios críticos do pipeline de solicitações do mapeador. Você pode implementar a [Interceptor](#) interface para implementar ganchos para observar ou modificar o processo do mapeador.

Você pode registrar um ou mais interceptores em um único DynamoDB Mapper como opção de configuração. Veja o [exemplo](#) no final desta seção para saber como registrar um interceptor.

Entenda o pipeline de solicitações

O pipeline de solicitações do mapeador consiste nas 5 etapas a seguir:

1. Inicialização: configure a operação e reúna o contexto inicial.
2. Serialização: converta objetos de solicitação de alto nível em objetos de solicitação de baixo nível. Essa etapa converte objetos Kotlin de alto nível em itens do DynamoDB que consistem em nomes e valores de atributos.
3. Invocação de baixo nível: execute uma solicitação no cliente subjacente do DynamoDB.
4. Desserialização: converta objetos de resposta de baixo nível em objetos de resposta de alto nível. Essa etapa inclui a conversão de itens do DynamoDB que consistem em nomes e valores de atributos em objetos Kotlin de alto nível.
5. Conclusão: finalize a resposta de alto nível para retornar ao chamador. Se uma exceção foi lançada durante a execução do pipeline, essa etapa finaliza a exceção que é lançada para o chamador.

Hooks

Ganchos são métodos interceptores que o mapeador invoca antes ou depois de etapas específicas no pipeline. Existem duas variantes de ganchos: somente leitura e modificação (ou leitura e gravação). Por exemplo, `readBeforeInvocation` é um gancho somente para leitura que o mapeador executa na fase anterior à etapa de invocação de baixo nível.

Ganchos somente para leitura

O mapeador invoca ganchos somente para leitura antes e depois de cada etapa no pipeline (exceto antes da etapa de inicialização e depois da etapa de conclusão). Os exaustores somente para leitura oferecem uma visão somente para leitura de uma operação de alto nível em andamento. Eles fornecem um mecanismo para examinar o estado de uma operação para registrar, depurar e

coletar métricas, por exemplo. Cada gancho somente para leitura recebe um argumento de contexto e retorna. [Unit](#)

O mapeador captura qualquer exceção lançada durante um gancho somente para leitura e a adiciona ao contexto. Em seguida, ele passa o contexto, com exceção dos ganchos interceptores subsequentes na mesma fase. O mapeador lança qualquer exceção para o chamador somente depois de chamar o gancho somente para leitura do último interceptor para a mesma fase. Por exemplo, se um mapeador estiver configurado com dois interceptores A e B o `readAfterSerialization` gancho A de `and` lançar uma exceção, o mapeador adiciona a exceção ao contexto passado para o gancho. B `readAfterSerialization` Depois B que o `readAfterSerialization` gancho for concluído, o mapeador devolve a exceção ao chamador.

Modificar ganchos

O mapeador invoca os ganchos de modificação antes de cada etapa no pipeline (exceto antes da inicialização). Os ganchos de modificação oferecem a capacidade de ver e modificar uma operação de alto nível em andamento. Eles podem ser usados para personalizar o comportamento e os dados de uma forma que a configuração do mapeador e os esquemas de itens não fazem. Cada gancho de modificação recebe um argumento de contexto e retorna algum subconjunto desse contexto como resultado — modificado pelo gancho ou transmitido do contexto de entrada.

Se o mapeador detectar alguma exceção ao executar um gancho de modificação, ele não executará nenhum gancho de modificação de outros interceptores na mesma fase. O mapeador adiciona a exceção ao contexto e a passa para o próximo gancho somente para leitura. O mapeador lança qualquer exceção para o chamador somente depois de chamar o gancho somente para leitura dos últimos interceptores para a mesma fase. Por exemplo, se um mapeador estiver configurado com dois interceptores A e B o `modifyBeforeSerialization` gancho A de `and` gerar uma exceção, B o `modifyBeforeSerialization` gancho não será invocado. AO `readAfterSerialization` gancho B ' dos interceptores e s será executado, após o qual a exceção será devolvida ao chamador.

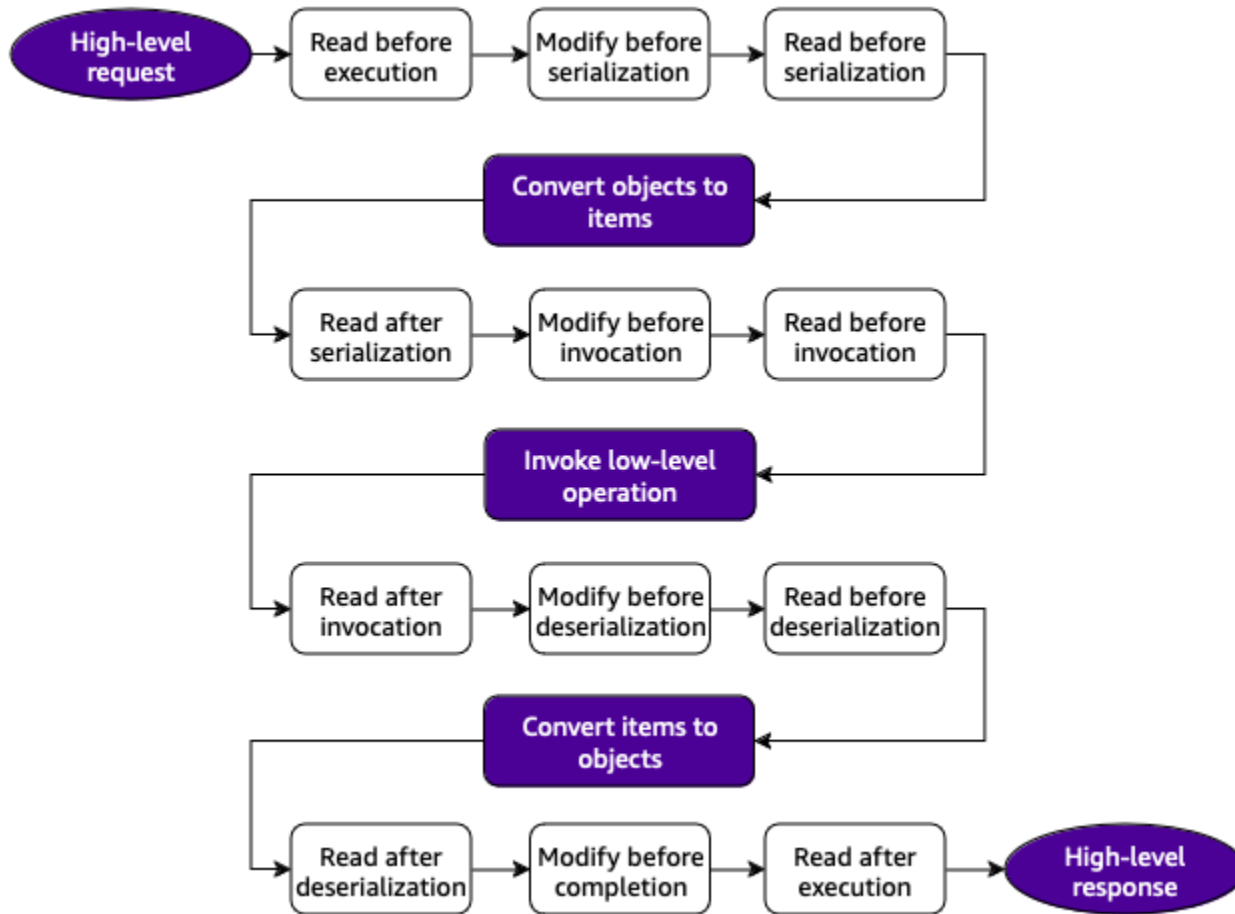
Ordem de execução

A ordem na qual os interceptores são definidos na configuração de um mapeador determina a ordem em que o mapeador chama os ganchos:

- Nas fases anteriores à etapa de invocação de baixo nível, ele executa os ganchos na mesma ordem em que foram adicionados na configuração.

- Para fases após a etapa de invocação de baixo nível, ele executa ganchos na ordem inversa da ordem em que foram adicionados na configuração.

O diagrama a seguir mostra a ordem de execução dos métodos de gancho:



Descrição em texto da ordem de execução dos métodos de gancho

Um mapeador executa os ganchos de um interceptor na seguinte ordem:

1. O DynamoDB Mapper invoca uma solicitação de alto nível
2. Leia antes da execução
3. Modificar antes da serialização
4. Leia antes da serialização
5. O DynamoDB Mapper converte objetos em itens
6. Leia após a serialização
7. Modificar antes da invocação

8. Leia antes da invocação
9. O DynamoDB Mapper invoca a operação de baixo nível
10. Leia após a invocação
11. Modificar antes da desserialização
12. Leia antes da desserialização
13. O DynamoDB Mapper converte itens em objetos
14. Leia após a desserialização
15. Modificar antes da conclusão
16. Leia após a execução
17. O DynamoDB Mapper retorna uma resposta de alto nível

Exemplo de configuração

O exemplo a seguir mostra como configurar um interceptor em uma `DynamoDbMapper` instância:


```
import aws.sdk.kotlin.hll.dynamodbmapper.DynamoDbMapper
import aws.sdk.kotlin.hll.dynamodbmapper.operations.ScanRequest
import aws.sdk.kotlin.hll.dynamodbmapper.operations.ScanResponse
import aws.sdk.kotlin.hll.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
}
```


Gere um esquema a partir de anotações

 O DynamoDB Mapper é uma versão prévia para desenvolvedores. O recurso não está completo e está sujeito a alterações.

O DynamoDB Mapper depende de esquemas que definem o mapeamento entre suas classes do Kotlin e os itens do DynamoDB. Suas classes Kotlin podem impulsionar a criação de esquemas usando o plug-in Gradle do gerador de esquemas.

Aplique o plugin

Para começar a gerar esquemas de código para suas classes, aplique o plug-in no script de construção do seu aplicativo e adicione uma dependência no módulo de anotações. O seguinte trecho de script do Gradle mostra a configuração necessária para a geração de código.

(Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.)

```
// 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")
}
```

Configurar o plug-in

O plug-in oferece várias opções de configuração que você pode aplicar usando a extensão do `dynamoDbMapper { ... }` plug-in em seu script de construção:

Opção	Descrição da opção	Valores
<code>generateBuilderClasses</code>	Controla se as classes DSL -style builder serão geradas	WHEN_REQUIRED (padrão): As classes Builder não

Opção	Descrição da opção	Valores
	para classes anotadas com <code>@DynamoDbItem</code>	serão geradas para classes que consistem apenas em membros mutáveis públicos e tenham um construtor de argumento zero ALWAYS: As classes Builder sempre serão geradas
<code>visibility</code>	Controla a visibilidade das classes geradas	PUBLIC (padrão) INTERNAL
<code>destinationPackage</code>	Especifica o nome do pacote para as classes geradas	RELATIVE(padão): as classes do esquema serão geradas em um subpacote relativo à sua classe anotada. Por padrão, o subpacote é nomeado <code>dynamodbapper.generatedschemas</code> e isso é configurável passando um parâmetro de string ABSOLUTE: as classes de esquema serão geradas em um pacote absoluto em relação à raiz do seu aplicativo. Por padrão, o pacote é nomeado <code>aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas</code> e isso pode ser configurado passando um parâmetro de string.

Opção	Descrição da opção	Valores
<code>generateGetTableExtension</code>	Controla se um método <code>DynamoDbMapper.getTable</code> de extensão será gerado	<code>true</code> (padrão) <code>false</code>

Example Exemplo de configuração de plug-in de geração de código

O exemplo a seguir configura o pacote de destino e a visibilidade do esquema gerado:

```
// 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
}
```

Anote as aulas

O gerador de esquemas procura anotações de classe para determinar para quais classes gerar esquemas. Para optar por gerar esquemas, anote suas classes com [@DynamoDbItem](#). Você também deve anotar uma propriedade de classe que serve como chave de partição do item com a [@DynamoDbPartitionKey](#) anotação.

A definição de classe a seguir mostra as anotações minimamente necessárias para a geração do esquema:

Example

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

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

Anotações de classe

As anotações a seguir são aplicadas às classes para controlar a geração do esquema:

- `@DynamoDbItem`: especifica que essa classe/interface descreve um tipo de item em uma tabela. Todas as propriedades públicas desse tipo serão mapeadas para atributos, a menos que sejam explicitamente ignoradas. Quando presente, um esquema será gerado para essa classe.
- `converterName`: um parâmetro opcional que indica que um esquema personalizado deve ser usado em vez daquele criado pelo plug-in gerador de esquema. Esse é o nome totalmente qualificado da `ItemConverter` classe personalizada. A [the section called “Definir um conversor de itens personalizado”](#) seção mostra um exemplo de criação e uso de um esquema personalizado.

Anotações de propriedade

Você pode aplicar as seguintes anotações às propriedades da classe para controlar a geração do esquema:

- `@DynamoDbPartitionKey`: especifica a chave de partição do item.
- `@DynamoDbSortKey`: especifica uma chave de classificação opcional para o item.
- `@DynamoDbIgnore`: especifica que essa propriedade de classe não deve ser convertida de/para um atributo Item pelo DynamoDB Mapper.
- `@DynamoDbAttribute`: especifica um nome de atributo personalizado opcional para essa propriedade de classe.

Definir um conversor de itens personalizado

Em alguns casos, talvez você queira definir um conversor de itens personalizado para sua classe. Uma razão para isso seria se sua classe usasse um tipo que não é suportado pelo plug-in gerador de esquema. Usamos a seguinte versão da `Employee` classe como exemplo:

```
import kotlin.uuid.Uuid  
  
@DynamoDbItem
```

```
data class Employee(  
    @DynamoDbPartitionKey  
    var id: Int,  
  
    var name: String,  
    var role: String,  
    var workstationId: Uuid  
)
```

A `Employee` classe agora usa um `kotlin.uuid.Uuid` tipo, que atualmente não é suportado pelo gerador de esquema. A geração do esquema falha com um erro: `Unsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)`. Esse erro indica que o plug-in não pode gerar um conversor de itens para essa classe. Portanto, precisamos escrever nossos próprios.

Para fazer isso, implementamos um [ItemConverter](#) para a classe e, em seguida, modificamos a anotação da `@DynamoDbItem` classe especificando o nome totalmente qualificado do novo conversor de itens.

Primeiro, implementamos um [ValueConverter](#) para a `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())  
}
```

Em seguida, implementamos um `ItemConverter` para nossa `Employee` classe. O `ItemConverter` usa esse novo conversor de valor no descritor de atributo para "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
        )
    ),
)
```

Agora que definimos o conversor de itens, podemos aplicá-lo à nossa classe. Atualizamos a `@DynamoDbItem` anotação para referenciar o conversor de itens fornecendo o nome da classe totalmente qualificado, conforme mostrado a seguir:

```
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  
)
```

Finalmente, podemos começar a usar a classe com o DynamoDB Mapper.

Defina esquemas manualmente

⚠ O DynamoDB Mapper é uma versão prévia para desenvolvedores. O recurso não está completo e está sujeito a alterações.

Definir um esquema no código

Para obter o máximo de controle e personalização, você pode definir e personalizar manualmente os esquemas no código.

Conforme mostrado no trecho a seguir, você precisa incluir menos dependências em seu `build.gradle.kts` arquivo em comparação com o uso da criação de esquemas orientados por anotações.

(Você pode navegar até o [X.Y.Z](#) link para ver a versão mais recente disponível.)

```
// 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.  
}
```

Observe que você não precisa do plug-in gerador de esquemas nem do pacote de anotações.

O mapeamento entre uma classe Kotlin e um item do DynamoDB requer uma [ItemSchema<T>](#) implementação, onde T está o tipo da classe Kotlin. Um esquema consiste nos seguintes elementos:

- Um conversor de itens, que define como converter entre instâncias de objetos Kotlin e itens do DynamoDB.

- Uma especificação de chave de partição, que define o nome e o tipo do atributo da chave de partição.
- Opcionalmente, uma especificação de chave de classificação, que define o nome e o tipo do atributo da chave de classificação.

No código a seguir, criamos manualmente uma `CarSchema` instância:

```
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)
```

O código anterior cria um conversor chamado `carConverter`, que é definido como uma implementação anônima de `ItemConverter<Car>`. O `convertTo` método do conversor aceita

um `Car` argumento e retorna uma `Item` instância representando as chaves e valores literais dos atributos de itens do DynamoDB. O `convertFrom` método do conversor aceita um `Item` argumento e retorna uma `Car` instância dos valores dos atributos do `Item` argumento.

Em seguida, o código cria duas especificações principais: uma para a chave de partição e outra para a chave de classificação. Cada tabela ou índice do DynamoDB deve ter exatamente uma chave de partição e, da mesma forma, cada definição de esquema do DynamoDB Mapper. Os esquemas também podem ter uma chave de classificação.

Na última declaração, o código cria um esquema para a tabela do `cars` DynamoDB a partir do conversor e das principais especificações.

O esquema resultante é equivalente ao esquema baseado em anotações que geramos na seção. [the section called “Defina um esquema com anotações de classe”](#) Para referência, a seguir está a classe anotada que usamos:

Classe `Car` com anotações do DynamoDB Mapper

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

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

Além de implementar sua própria `ItemConverter`, o DynamoDB Mapper inclui várias implementações úteis, como:

- [SimpleItemConverter](#): fornece uma lógica de conversão simples usando uma classe construtora e descritores de atributos. Veja o exemplo em [the section called “Definir um conversor de itens personalizado”](#) para saber como você pode usar essa implementação.
- [HeterogeneousItemConverter](#): fornece lógica de conversão de tipos polimórficos usando um atributo discriminador e instâncias delegadas `ItemConverter` para subtipos.
- [DocumentConverter](#): fornece lógica de conversão para dados não estruturados em [Document](#) objetos.

Use índices secundários com o DynamoDB Mapper

⚠ O DynamoDB Mapper é uma versão prévia para desenvolvedores. O recurso não está completo e está sujeito a alterações.

Definir um esquema para um índice secundário

As tabelas do DynamoDB oferecem suporte a índices secundários que fornecem acesso aos dados usando chaves diferentes das definidas na própria tabela base. Assim como nas tabelas base, o DynamoDB Mapper interage com os índices usando o tipo. [ItemSchema](#)

Os índices secundários do DynamoDB não precisam conter todos os atributos da tabela base. Assim, a classe Kotlin que mapeia para um índice pode ser diferente da classe Kotlin que mapeia para a tabela base desse índice. Quando for esse o caso, um esquema separado deve ser declarado para a classe de índice.

O código a seguir cria manualmente um esquema de índice para a tabela do DynamoDB. `cars`

```
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.h11.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
        )
    */

```

Agora podemos usar `Model` instâncias em operações.

Use índices secundários nas operações

O DynamoDB Mapper suporta um subconjunto de operações em índices, a saber e. `queryPaginated` `scanPaginated` Para invocar essas operações em um índice, você deve primeiro obter uma referência a um índice do objeto de tabela. No exemplo a seguir, usamos o `modelSchema` que criamos anteriormente para o `cars-by-model` índice (criação não mostrada aqui):

```

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

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

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

```

Use expressões

⚠ O DynamoDB Mapper é uma versão prévia para desenvolvedores. O recurso não está completo e está sujeito a alterações.

Algumas operações do DynamoDB [aceitam](#) expressões que você pode usar para especificar restrições ou condições. O DynamoDB Mapper fornece um Kotlin idiomático para criar expressõesDSL. DSLIsso traz maior estrutura e legibilidade ao seu código e também facilita a escrita de expressões.

Esta seção descreve a DSL sintaxe e fornece vários exemplos.

Use expressões em operações

Você usa expressões em operações como `scan`, onde elas filtram os itens retornados com base nos critérios que você define. Para usar expressões com o DynamoDB Mapper, adicione o componente de expressão na solicitação de operação.

O trecho a seguir mostra um exemplo de uma expressão de filtro usada em uma `scan` operação. Ele usa um argumento `lambda` para descrever os critérios de filtro que limitam os itens a serem retornados àqueles com um valor de `year` atributo de 2001:

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

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

O exemplo a seguir mostra uma `query` operação que oferece suporte a expressões em dois lugares: filtragem por chave de classificação e filtragem sem chave:

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

```
}
```

O código anterior filtra os resultados para aqueles que atendem aos três critérios:

- O valor do atributo da chave de partição é 1000 - AND -
- O valor do atributo chave de classificação começa com a letra M - AND -
- o valor do atributo do ano é 2001

Componentes do DSL

A DSL sintaxe expõe vários tipos de componentes, descritos abaixo, que você usa para criar expressões.

Atributos

A maioria das condições faz referência a atributos, que são identificados por sua chave ou caminho do documento. Com o DSK, você cria todas as referências de atributos usando a `attr` função e, opcionalmente, faz modificações adicionais.

O código a seguir mostra uma variedade de exemplos de referências de atributos, do simples ao complexo, como desreferenciamento de listas por índice e desreferenciamento de mapas por chave:

```
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`.
```

Igualdades e desigualdades

Você pode comparar valores de atributos em uma expressão por igualdades e desigualdades. Você pode comparar valores de atributos com valores literais ou outros valores de atributos. As funções que você usa para especificar as condições são:

- `eq`: é igual a (equivalente `a==`)
- `neq`: não é igual a (equivalente `a!=`)
- `gt`: é maior que (equivalente `a>`)

- `gte`: é maior ou igual a (equivalente a `>=`)
- `lt`: é menor que (equivalente a `<`)
- `lte`: é menor ou igual a (equivalente a `<=`)

Você combina a função de comparação com argumentos usando a notação infixa, conforme mostrado nos exemplos a seguir:

```
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`.
```

Intervalos e conjuntos

Além dos valores únicos, você pode comparar valores de atributos com vários valores em intervalos ou conjuntos. Você usa a [isIn](#) função infix para fazer a comparação, conforme mostrado nos exemplos a seguir:

```
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",
)
```

A `isIn` função fornece sobrecargas para coleções (como `Set<String>`) e para limites que você pode expressar como um Kotlin [ClosedRange<T>](#) (como). [IntRange](#) Para limites que você não pode expressar como a `ClosedRange<T>` (como matrizes de bytes ou outras referências de atributos), você pode usar a função: [isBetween](#)

```
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`.
```

Lógica booleana

Você pode combinar condições individuais ou alteradas usando a lógica booleana usando as seguintes funções:

- **and**: toda condição deve ser verdadeira (equivalente a `&&`)
- **or**: pelo menos uma condição deve ser verdadeira (equivalente a `||`)
- **not**: a condição dada deve ser falsa (equivalente a `!`)

Os exemplos a seguir mostram cada função:

```
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.
)
```

Você também pode combinar condições booleanas por meio de funções booleanas para criar uma lógica aninhada, conforme mostrado na expressão a seguir:

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

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

A expressão anterior filtra os resultados para aqueles que atendem a uma dessas condições:

- Ambas as condições são verdadeiras:
 - fooo valor do atributo é 123 - AND -
 - baro valor do atributo é "abc"
- Ambas as condições são verdadeiras:
 - fooo valor do atributo é 234 - - AND
 - baro valor do atributo é "bcd"

Isso é equivalente à seguinte expressão booleana Kotlin:

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

Funções e propriedades

As funções e propriedades a seguir fornecem recursos adicionais de expressão:

- [contains](#): verifica se um valor de atributo de string/lista contém um determinado valor
- [exists](#): verifica se um atributo está definido e contém algum valor (incluindo null)
- [notExists](#): verifica se um atributo é indefinido
- [isOfType](#): verifica se um valor de atributo é de um determinado tipo, como string, número, booleano e assim por diante
- [size](#): obtém o tamanho de um atributo, como o número de elementos em uma coleção ou o comprimento de uma string
- [startsWith](#): verifica se o valor de um atributo de string começa com uma determinada substring

Os exemplos a seguir mostram o uso de funções e propriedades adicionais que você pode usar em expressões:

```
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.
```

Filtros de chave de classificação

As expressões de filtro nas chaves de classificação (como no `keyCondition` parâmetro da query operação) não usam valores de atributos nomeados. Para usar uma chave de classificação em um filtro, você deve usar a palavra-chave `sortKey` em todas as comparações. A `sortKey` palavra-chave substitui `attr("<sort key name>")` conforme mostrado nos exemplos a seguir:

```
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).
```

Você não pode combinar filtros de chave de classificação com a lógica booleana e eles suportam somente um subconjunto das comparações descritas acima:

- [Igualdades e desigualdades](#): todas as comparações são suportadas
- [Intervalos e conjuntos](#): todas as comparações são suportadas
- [Lógica booleana](#): não suportada
- [Funções e propriedades](#): somente `startsWith` é suportado

Exemplos de código SDK para Kotlin

Os exemplos de código neste tópico mostram como usar o AWS SDK para Kotlin com. AWS

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como aproveitar bibliotecas ou funções específicas do serviço.

Serviços

- [Exemplos de API Gateway usando SDK para Kotlin](#)
- [Exemplos de Aurora usando o SDK para Kotlin](#)
- [Exemplos do Auto Scaling usando o SDK para Kotlin](#)
- [Exemplos do Amazon Bedrock usando o SDK para Kotlin](#)
- [CloudWatch exemplos usando SDK para Kotlin](#)
- [CloudWatch Exemplos de registros usando o SDK para Kotlin](#)
- [Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Kotlin](#)
- [Exemplos do Amazon Comprehend usando o SDK para Kotlin](#)
- [Exemplos do DynamoDB usando o SDK para Kotlin](#)
- [EC2 Exemplos da Amazon usando SDK para Kotlin](#)
- [Exemplos do Amazon ECR usando SDK para Kotlin](#)
- [OpenSearch Exemplos de serviços usando o SDK para Kotlin](#)
- [EventBridge exemplos usando SDK para Kotlin](#)
- [AWS Glue exemplos usando SDK para Kotlin](#)
- [Exemplos de IAM usando o SDK para Kotlin](#)
- [AWS IoT exemplos usando SDK para Kotlin](#)
- [AWS IoT data exemplos usando SDK para Kotlin](#)

- [Exemplos do Amazon Keyspaces usando o SDK para Kotlin](#)
- [AWS KMS exemplos usando SDK para Kotlin](#)
- [Exemplos de Lambda usando o SDK para Kotlin](#)
- [MediaConvert exemplos usando SDK para Kotlin](#)
- [Exemplos do Amazon Pinpoint usando o SDK para Kotlin](#)
- [Exemplos do Amazon RDS usando o SDK para Kotlin](#)
- [Exemplos de serviços de dados do Amazon RDS usando SDK para Kotlin](#)
- [Exemplos do Amazon Redshift usando o SDK para Kotlin](#)
- [Exemplos do Amazon Rekognition usando o SDK para Kotlin](#)
- [Exemplos de registro de domínios do Route 53 usando SDKs para Kotlin](#)
- [Exemplos do Amazon S3 usando o SDK para Kotlin](#)
- [SageMaker Exemplos de IA usando SDK para Kotlin](#)
- [Exemplos de Secrets Manager usando o SDK para Kotlin](#)
- [Exemplos do Amazon SES usando SDK para Kotlin](#)
- [Exemplos do Amazon SNS usando o SDK para Kotlin](#)
- [Exemplos do Amazon SQS usando o SDK para Kotlin](#)
- [Exemplos do Step Functions usando o SDK para Kotlin](#)
- [Suporte exemplos usando SDK para Kotlin](#)
- [Exemplos do Amazon Translate usando SDK para Kotlin](#)

Exemplos de API Gateway usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o API Gateway.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Cenários](#)

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos de Aurora usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com Aurora.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)
- [Cenários](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de cluster do banco de dados do Aurora e definir os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Criar um snapshot do cluster do banco de dados e limpar os recursos.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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(s1Time * 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(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!")
}

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}")
    }
}
}

```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CriarDBCluster](#)
 - [CriarDBClusterParameterGroup](#)
 - [Criar DBCluster instantâneo](#)
 - [CriarDBInstance](#)
 - [ExcluirDBCluster](#)
 - [ExcluirDBClusterParameterGroup](#)
 - [ExcluirDBInstance](#)
 - [DescreverDBClusterParameterGroups](#)
 - [Descreva DBCluster os parâmetros](#)
 - [Descreva os DBCluster instantâneos](#)
 - [DescreverDBClusters](#)
 - [Descreva DBEngine as versões](#)
 - [DescreverDBInstances](#)
 - [DescribeOrderableDBInstanceOpções](#)

Ações

CreateDBCluster

O código de exemplo a seguir mostra como usar CreateDBCluster.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte [Criar DBCluster](#) no AWS SDK para referência da API Kotlin.

CreateDBClusterParameterGroup

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte [Criar DBCluster ParameterGroup](#) no AWS SDK para referência da API Kotlin.

CreateDBClusterSnapshot

O código de exemplo a seguir mostra como usar `CreateDBClusterSnapshot`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte [Criar DBCluster instantâneo](#) no AWS SDK para referência da API Kotlin.

CreateDBInstance

O código de exemplo a seguir mostra como usar CreateDBInstance.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte [Criar DBInstance](#) no AWS SDK para referência da API Kotlin.

DeleteDBCluster

O código de exemplo a seguir mostra como usar DeleteDBCluster.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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!")
    }
}

```

- Para obter detalhes da API, consulte [Excluir DBCluster](#) no AWS SDK para referência da API Kotlin.

DeleteDBClusterParameterGroup

O código de exemplo a seguir mostra como usar DeleteDBClusterParameterGroup.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}


```

- Para obter detalhes da API, consulte [Excluir DBCluster ParameterGroup](#) no AWS SDK para referência da API Kotlin.

DeleteDBInstance

O código de exemplo a seguir mostra como usar DeleteDBInstance.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte [Excluir DBInstance](#) no AWS SDK para referência da API Kotlin.

DescribeDBClusterParameterGroups

O código de exemplo a seguir mostra como usar DescribeDBClusterParameterGroups.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
}
```

- Para obter detalhes da API, consulte [Descrver DBCluster ParameterGroups](#) no AWS SDK para referência da API Kotlin.

DescribeDBClusterParameters

O código de exemplo a seguir mostra como usar DescribeDBClusterParameters.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
                    }
                }
            }
        }
    }
}


```

- Para obter detalhes da API, consulte [Descrição DBCluster dos parâmetros](#) no AWS SDK para referência da API Kotlin.

DescribeDBClusterSnapshots

O código de exemplo a seguir mostra como usar DescribeDBClusterSnapshots.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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(1Time * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Para obter detalhes da API, consulte [Descrever DBCluster instantâneos](#) no AWS SDK para referência da API Kotlin.

DescribeDBClusters

O código de exemplo a seguir mostra como usar DescribeDBClusters.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
}
```

- Para obter detalhes da API, consulte [Descrever DBClusters](#) no AWS SDK para referência da API Kotlin.

DescribeDBEngineVersions

O código de exemplo a seguir mostra como usar DescribeDBEngineVersions.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

```
    }  
  }  
}
```

- Para obter detalhes da API, consulte [Descrever DBEngine versões](#) no AWS SDK para referência da API Kotlin.

DescribeDBInstances

O código de exemplo a seguir mostra como usar DescribeDBInstances.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}

```

- Para obter detalhes da API, consulte [Descrver DBInstances](#) no AWS SDK para referência da API Kotlin.

ModifyDBClusterParameterGroup

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte [Modificar DBCluster ParameterGroup](#) no AWS SDK para referência da API Kotlin.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. [GitHub](#)

Serviços utilizados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Auto Scaling usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com Auto Scaling.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um grupo do Amazon EC2 Auto Scaling com um modelo de lançamento e zonas de disponibilidade e obtenha informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas de CloudWatch métricas e, em seguida, limpe os recursos.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.waitUntilGroupExists(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.waitUntilGroupExists(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")
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Ações

CreateAutoScalingGroup

O código de exemplo a seguir mostra como usar CreateAutoScalingGroup.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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!")
        }
    }
}
```

- Para obter detalhes da API, consulte a [CreateAutoScalingGroup](#) preferência da API AWS SDK for Kotlin.

DeleteAutoScalingGroup

O código de exemplo a seguir mostra como usar DeleteAutoScalingGroup.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteAutoScalingGroup](#) preferência da API AWS SDK for Kotlin.

DescribeAutoScalingGroups

O código de exemplo a seguir mostra como usar DescribeAutoScalingGroups.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
                }
            }
        }
    }
}

```

- Para obter detalhes da API, consulte a [DescribeAutoScalingGroups](#) referência da API AWS SDK for Kotlin.

DescribeAutoScalingInstances

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeAutoScalingInstances](#) referência da API AWS SDK for Kotlin.

DescribeScalingActivities

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeScalingActivities](#) referência da API AWS SDK for Kotlin.

DisableMetricsCollection

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DisableMetricsCollection](#) referência da API AWS SDK for Kotlin.

EnableMetricsCollection

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [EnableMetricsCollection](#) referência da API AWS SDK for Kotlin.

SetDesiredCapacity

O código de exemplo a seguir mostra como usar SetDesiredCapacity.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
}
```

- Para obter detalhes da API, consulte a [SetDesiredCapacity](#) referência da API AWS SDK for Kotlin.

TerminateInstanceInAutoScalingGroup

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```


- Para obter detalhes da API, consulte a [TerminateInstanceInAutoScalingGroup](#) preferência da API AWS SDK for Kotlin.

UpdateAutoScalingGroup

O código de exemplo a seguir mostra como usar UpdateAutoScalingGroup.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
```

```
}  
}
```

- Para obter detalhes da API, consulte a [UpdateAutoScalingGroup](#) referência da API AWS SDK for Kotlin.

Exemplos do Amazon Bedrock usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Ações](#)

Ações

ListFoundationModels

O código de exemplo a seguir mostra como usar ListFoundationModels.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Listar os modelos de base do Amazon Bedrock disponíveis.

```
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
    }
}

```

- Para obter detalhes da API, consulte a [ListFoundationModels](#) referência da API AWS SDK for Kotlin.

CloudWatch exemplos usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com CloudWatch

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Olá CloudWatch

O exemplo de código a seguir mostra como começar a usar o CloudWatch.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListMetrics](#) referência da API AWS SDK for Kotlin.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)


Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Listar CloudWatch namespaces e métricas.
- Obter estatísticas para uma métrica e para faturamento estimado.
- Criar e atualizar um painel.
- Criar e adicionar dados a uma métrica.
- Criar e acionar um alarme e, em seguida, visualizar o histórico de alarmes.
- Criar um detector de anomalias.
- Obter uma imagem de métrica e, em seguida, limpar os recursos.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute um cenário interativo demonstrando CloudWatch recursos.

```
/**  
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
}

```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [DeleteAlarms](#)

- [DeleteAnomalyDetector](#)
- [DeleteDashboards](#)
- [DescribeAlarmHistory](#)
- [DescribeAlarms](#)
- [DescribeAlarmsForMetric](#)
- [DescribeAnomalyDetectors](#)
- [GetMetricData](#)
- [GetMetricStatistics](#)
- [GetMetricWidgetImage](#)
- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

Ações

DeleteAlarms

O código de exemplo a seguir mostra como usar DeleteAlarms.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
}
```

- Para obter detalhes da API, consulte a [DeleteAlarms](#) referência da API AWS SDK for Kotlin.

DeleteAnomalyDetector

O código de exemplo a seguir mostra como usar DeleteAnomalyDetector.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteAnomalyDetector](#) referência da API AWS SDK for Kotlin.

DeleteDashboards

O código de exemplo a seguir mostra como usar DeleteDashboards.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteDashboards](#) referência da API AWS SDK for Kotlin.

DescribeAlarmHistory

O código de exemplo a seguir mostra como usar DescribeAlarmHistory.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeAlarmHistory](#) referência da API AWS SDK for Kotlin.

DescribeAlarms

O código de exemplo a seguir mostra como usar DescribeAlarms.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeAlarms](#) referência da API AWS SDK for Kotlin.

DescribeAlarmsForMetric

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeAlarmsForMetric](#) referência da API AWS SDK for Kotlin.

DescribeAnomalyDetectors

O código de exemplo a seguir mostra como usar DescribeAnomalyDetectors.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

```
    }  
  }  
}
```

- Para obter detalhes da API, consulte a [DescribeAnomalyDetectors](#) referência da API AWS SDK for Kotlin.

DisableAlarmActions

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")  
    }  
}
```

- Para obter detalhes da API, consulte a [DisableAlarmActions](#) referência da API AWS SDK for Kotlin.

EnableAlarmActions

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [EnableAlarmActions](#) referência da API AWS SDK for Kotlin.

GetMetricData

O código de exemplo a seguir mostra como usar GetMetricData.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}

```

- Para obter detalhes da API, consulte a [GetMetricData](#) referência da API AWS SDK for Kotlin.

GetMetricStatistics

O código de exemplo a seguir mostra como usar `GetMetricStatistics`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
        }
    }
}
}
}

```

- Para obter detalhes da API, consulte a [GetMetricStatistics](#) referência da API AWS SDK for Kotlin.

GetMetricWidgetImage

O código de exemplo a seguir mostra como usar `GetMetricWidgetImage`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
```

- Para obter detalhes da API, consulte a [GetMetricWidgetImage](#) referência da API AWS SDK for Kotlin.

ListDashboards

O código de exemplo a seguir mostra como usar ListDashboards.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
    }
}
```

- Para obter detalhes da API, consulte a [ListDashboards](#) referência da API AWS SDK for Kotlin.

ListMetrics

O código de exemplo a seguir mostra como usar ListMetrics.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
}
```

- Para obter detalhes da API, consulte a [ListMetrics](#) referência da API AWS SDK for Kotlin.

PutAnomalyDetector

O código de exemplo a seguir mostra como usar PutAnomalyDetector.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [PutAnomalyDetector](#) referência da API AWS SDK for Kotlin.

PutDashboard

O código de exemplo a seguir mostra como usar PutDashboard.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
}
```

- Para obter detalhes da API, consulte a [PutDashboard](#) referência da API AWS SDK for Kotlin.

PutMetricAlarm

O código de exemplo a seguir mostra como usar PutMetricAlarm.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}

```

- Para obter detalhes da API, consulte a [PutMetricAlarm](#) referência da API AWS SDK for Kotlin.

PutMetricData

O código de exemplo a seguir mostra como usar PutMetricData.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
}
```

- Para obter detalhes da API, consulte a [PutMetricData](#) referência da API AWS SDK for Kotlin.

CloudWatch Exemplos de registros usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin with Logs. CloudWatch

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Ações](#)

Ações

DeleteSubscriptionFilter

O código de exemplo a seguir mostra como usar DeleteSubscriptionFilter.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
}
```

- Para obter detalhes da API, consulte a [DeleteSubscriptionFilter](#) referência da API AWS SDK for Kotlin.

DescribeSubscriptionFilters

O código de exemplo a seguir mostra como usar DescribeSubscriptionFilters.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeSubscriptionFilters](#) referência da API AWS SDK for Kotlin.

StartLiveTail

O código de exemplo a seguir mostra como usar `StartLiveTail`.

SDK para Kotlin

Inclua os arquivos necessários.

```
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
```

Inicie a sessão do 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)
}
```

- Para obter detalhes da API, consulte a [StartLiveTail](#) referência da API AWS SDK for Kotlin.

Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

AdminGetUser

O código de exemplo a seguir mostra como usar AdminGetUser.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [AdminGetUser](#) referência da API AWS SDK for Kotlin.

AdminInitiateAuth

O código de exemplo a seguir mostra como usar AdminInitiateAuth.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [AdminInitiateAuth](#) referência da API AWS SDK for Kotlin.

AdminRespondToAuthChallenge

O código de exemplo a seguir mostra como usar AdminRespondToAuthChallenge.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 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}")
    }
}
```

- Para obter detalhes da API, consulte a [AdminRespondToAuthChallenge](#) referência da API AWS SDK for Kotlin.

AssociateSoftwareToken

O código de exemplo a seguir mostra como usar AssociateSoftwareToken.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
}
}
```

- Para obter detalhes da API, consulte a [AssociateSoftwareToken](#) referência da API AWS SDK for Kotlin.

ConfirmSignUp

O código de exemplo a seguir mostra como usar ConfirmSignUp.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [ConfirmSignUp](#) preferência da API AWS SDK for Kotlin.

ListUsers

O código de exemplo a seguir mostra como usar ListUsers.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListUsers](#) referência da API AWS SDK for Kotlin.

ResendConfirmationCode

O código de exemplo a seguir mostra como usar ResendConfirmationCode.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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))
    }
}
```

- Para obter detalhes da API, consulte a [ResendConfirmationCode](#) referência da API AWS SDK for Kotlin.

SignUp

O código de exemplo a seguir mostra como usar SignUp.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
}
```

- Para obter detalhes da API, consulte a [SignUp](#) preferência da API AWS SDK for Kotlin.

VerifySoftwareToken

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
}
}
```

```
}
```

- Para obter detalhes da API, consulte a [VerifySoftwareToken](#) referência da API AWS SDK for Kotlin.

Cenários

Inscrever um usuário em um grupo de usuários que exija MFA

O exemplo de código a seguir mostra como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Fazer login usando uma senha e um código de MFA.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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("**** Confirmation 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>()

```

```
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}")
    }
}

// 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")
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)

- [ConfirmSignUp](#)
- [InitiateAuth](#)
- [ListUsers](#)
- [ResendConfirmationCode](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [VerifySoftwareToken](#)

Exemplos do Amazon Comprehend usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Comprehend.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Cenários](#)

Cenários

Crie um aplicativo de mensagem

O exemplo de código a seguir mostra como criar um aplicativo de mensagens usando o Amazon SQS.

SDK para Kotlin

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços utilizados neste exemplo

- Amazon Comprehend
- Amazon SQS

Exemplos do DynamoDB usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)
- [Cenários](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.

- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie uma tabela do 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}")
}
}

```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```

// 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.")
    }
}
```

Obtenha um item de uma tabela.

```
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)
        }
    }
}

```

Exemplo completo.

```

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]}")  
            }  
        }  
    }  
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [Query](#)
 - [Scan](#)
 - [UpdateItem](#)

Ações

CreateTable

O código de exemplo a seguir mostra como usar CreateTable.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}

```

- Para obter detalhes da API, consulte a [CreateTable](#) referência da API AWS SDK for Kotlin.

DeleteItem

O código de exemplo a seguir mostra como usar DeleteItem.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteItem](#) referência da API AWS SDK for Kotlin.

DeleteTable

O código de exemplo a seguir mostra como usar DeleteTable.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteTable](#) referência da API AWS SDK for Kotlin.

GetItem

O código de exemplo a seguir mostra como usar GetItem.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
        }
    }
}
```

- Para obter detalhes da API, consulte a [GetItem](#) referência da API AWS SDK for Kotlin.

ListTables

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListTables](#) referência da API AWS SDK for Kotlin.

PutItem

O código de exemplo a seguir mostra como usar PutItem.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [PutItem](#) referência da API AWS SDK for Kotlin.

Query

O código de exemplo a seguir mostra como usar Query.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}

```

- Para obter detalhes da API, consulte [Query](#) na Referência da API AWS SDK para Kotlin.

Scan

O código de exemplo a seguir mostra como usar Scan.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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]}")
        }
    }
}
}
```

- Para obter detalhes da API, consulte [Scan](#) na Referência da API AWS SDK para Kotlin.

UpdateItem

O código de exemplo a seguir mostra como usar UpdateItem.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
}
```

- Para obter detalhes da API, consulte a [UpdateItem](#) referência da API AWS SDK for Kotlin.

Cenários

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

O exemplo de código a seguir mostra como criar um aplicativo que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

SDK para Kotlin

Mostra como criar uma aplicação Android nativa que envia dados usando a API Kotlin do Amazon DynamoDB e envia uma mensagem de texto usando a API Kotlin do Amazon SNS.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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\",
        \\\nErnest 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")
    }
}
}

```

- Para obter detalhes da API, consulte a [BatchExecuteStatement](#) referência da API AWS SDK for Kotlin.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
}

```

- Para obter detalhes da API, consulte a [ExecuteStatement](#) referência da API AWS SDK for Kotlin.

EC2 Exemplos da Amazon usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com a Amazon. EC2

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Olá Amazon EC2

Os exemplos de código a seguir mostram como começar a usar a Amazon EC2.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeSecurityGroups](#) referência da API AWS SDK for Kotlin.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.waitForInstanceStopped {
            // 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")
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

Ações

AllocateAddress

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [AllocateAddress](#) referência da API AWS SDK for Kotlin.

AssociateAddress

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [AssociateAddress](#) referência da API AWS SDK for Kotlin.

AuthorizeSecurityGroupIngress

O código de exemplo a seguir mostra como usar AuthorizeSecurityGroupIngress.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```



```
}
```

- Para obter detalhes da API, consulte a [AuthorizeSecurityGroupIngress](#) referência da API AWS SDK for Kotlin.

CreateKeyPair

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [CreateKeyPair](#) referência da API AWS SDK for Kotlin.

CreateSecurityGroup

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
}
}
```

- Para obter detalhes da API, consulte a [CreateSecurityGroup](#) referência da API AWS SDK for Kotlin.

DeleteKeyPair

O código de exemplo a seguir mostra como usar DeleteKeyPair.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteKeyPair](#) referência da API AWS SDK for Kotlin.

DeleteSecurityGroup

O código de exemplo a seguir mostra como usar DeleteSecurityGroup.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteSecurityGroup](#) referência da API AWS SDK for Kotlin.

DescribeInstanceTypes

O código de exemplo a seguir mostra como usar DescribeInstanceTypes.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [DescribeInstanceTypes](#) referência da API AWS SDK for Kotlin.

DescribeInstances

O código de exemplo a seguir mostra como usar DescribeInstances.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeInstances](#) referência da API AWS SDK for Kotlin.

DescribeKeyPairs

O código de exemplo a seguir mostra como usar DescribeKeyPairs.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeKeyPairs](#) referência da API AWS SDK for Kotlin.

DescribeSecurityGroups

O código de exemplo a seguir mostra como usar DescribeSecurityGroups.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
}
```

- Para obter detalhes da API, consulte a [DescribeSecurityGroups](#) referência da API AWS SDK for Kotlin.

DisassociateAddress

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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!")
    }
}
```

- Para obter detalhes da API, consulte a [DisassociateAddress](#) referência da API AWS SDK for Kotlin.

ReleaseAddress

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [ReleaseAddress](#) referência da API AWS SDK for Kotlin.

RunInstances

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
        }
    }
}
```

- Para obter detalhes da API, consulte a [RunInstances](#) referência da API AWS SDK for Kotlin.

StartInstances

O código de exemplo a seguir mostra como usar StartInstances.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [StartInstances](#) referência da API AWS SDK for Kotlin.

StopInstances

O código de exemplo a seguir mostra como usar StopInstances.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

- Para obter detalhes da API, consulte a [StopInstances](#) referência da API AWS SDK for Kotlin.

TerminateInstances

O código de exemplo a seguir mostra como usar TerminateInstances.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [TerminateInstances](#) referência da API AWS SDK for Kotlin.

Exemplos do Amazon ECR usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon ECR.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Hello Amazon ECR

Os exemplos de código a seguir mostram como começar a usar o Amazon ECR.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte [listImages](#) na Referência de APIs do AWS SDK para Kotlin.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um repositório do Amazon ECR.
- Defina políticas de repositório.
- Recupere o repositório URLs.
- Obtenha tokens de autorização do Amazon ECR.
- Defina políticas de ciclo de vida para repositórios do Amazon ECR.
- Envie por push uma imagem do Docker para um repositório do Amazon ECR.
- Verifique a existência de uma imagem em um repositório do Amazon ECR.
- Liste os repositórios do Amazon ECR da conta e verifique os detalhes sobre eles.
- Exclua repositórios do Amazon ECR.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute um cenário interativo que demonstre os recursos do 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.")
        }
    }
}
}
```

Uma classe de wrapper para os métodos do SDK do 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"
            } ]
        }

        """.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.")
        }
    }
}

/**
 * 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)  
    }  
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CreateRepository](#)
 - [DeleteRepository](#)
 - [DescribeImages](#)
 - [DescribeRepositories](#)
 - [GetAuthorizationToken](#)
 - [GetRepositoryPolicy](#)
 - [SetRepositoryPolicy](#)
 - [StartLifecyclePolicyPreview](#)

Ações

CreateRepository

O código de exemplo a seguir mostra como usar `CreateRepository`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [CreateRepository](#) referência da API AWS SDK for Kotlin.

DeleteRepository

O código de exemplo a seguir mostra como usar DeleteRepository.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteRepository](#) referência da API AWS SDK for Kotlin.

DescribeImages

O código de exemplo a seguir mostra como usar DescribeImages.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
        }
    }
}
```

```

    }
  }
}

```

- Para obter detalhes da API, consulte a [DescribeImages](#) referência da API AWS SDK for Kotlin.

DescribeRepositories

O código de exemplo a seguir mostra como usar DescribeRepositories.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 ""
    }
}
}
```

- Para obter detalhes da API, consulte a [DescribeRepositories](#) referência da API AWS SDK for Kotlin.

GetAuthorizationToken

O código de exemplo a seguir mostra como usar GetAuthorizationToken.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
        }
    }
}
```

- Para obter detalhes da API, consulte a [GetAuthorizationToken](#) referência da API AWS SDK for Kotlin.

GetRepositoryPolicy

O código de exemplo a seguir mostra como usar `GetRepositoryPolicy`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [GetRepositoryPolicy](#) referência da API AWS SDK for Kotlin.

PushImageCmd

O código de exemplo a seguir mostra como usar PushImageCmd.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
}
}

```

- Para obter detalhes da API, consulte a [PushImageCmd](#) referência da API AWS SDK for Kotlin.

SetRepositoryPolicy

O código de exemplo a seguir mostra como usar SetRepositoryPolicy.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
        }
    }
}
```

- Para obter detalhes da API, consulte a [SetRepositoryPolicy](#) referência da API AWS SDK for Kotlin.

StartLifecyclePolicyPreview

O código de exemplo a seguir mostra como usar StartLifecyclePolicyPreview.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
        }
    }
}
```

```
    }  
  }  
}
```

- Para obter detalhes da API, consulte a [StartLifecyclePolicyPreview](#) referência da API AWS SDK for Kotlin.

OpenSearch Exemplos de serviços usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin with Service. OpenSearch

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Ações](#)

Ações

CreateDomain

O código de exemplo a seguir mostra como usar CreateDomain.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
}
}
```

- Para obter detalhes da API, consulte a [CreateDomain](#) referência da API AWS SDK for Kotlin.

DeleteDomain

O código de exemplo a seguir mostra como usar DeleteDomain.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteDomain](#) referência da API AWS SDK for Kotlin.

ListDomainNames

O código de exemplo a seguir mostra como usar ListDomainNames.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
}
```

- Para obter detalhes da API, consulte a [ListDomainNames](#) referência da API AWS SDK for Kotlin.

UpdateDomainConfig

O código de exemplo a seguir mostra como usar UpdateDomainConfig.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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())
    }
}
```

- Para obter detalhes da API, consulte a [UpdateDomainConfig](#) referência da API AWS SDK for Kotlin.

EventBridge exemplos usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. EventBridge

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Olá EventBridge

O exemplo de código a seguir mostra como começar a usar o EventBridge.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListEventBuses](#) referência da API AWS SDK for Kotlin.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)


Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma regra e adicionar um destino a ela.
- Habilitar e desabilitar regras.
- Listar e atualizar regras e destinos.
- Enviar eventos e, em seguida, limpar os recursos.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 delObj =
    Delete {
        objects = toDelete
    }

val dor =
    DeleteObjectsRequest {
        bucket = bucketName
        delete = delObj
    }
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 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.")
    }
}

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
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [DeleteRule](#)
 - [DescribeRule](#)
 - [DisableRule](#)
 - [EnableRule](#)
 - [ListRuleNamesByTarget](#)
 - [ListRules](#)
 - [ListTargetsByRule](#)
 - [PutEvents](#)
 - [PutRule](#)
 - [PutTargets](#)

Ações

DeleteRule

O código de exemplo a seguir mostra como usar DeleteRule.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteRule](#) referência da API AWS SDK for Kotlin.

DescribeRule

O código de exemplo a seguir mostra como usar DescribeRule.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DescribeRule](#) referência da API AWS SDK for Kotlin.

DisableRule

O código de exemplo a seguir mostra como usar DisableRule.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
        }
    }
}
```

- Para obter detalhes da API, consulte a [DisableRule](#) referência da API AWS SDK for Kotlin.

EnableRule

O código de exemplo a seguir mostra como usar EnableRule.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
        }
    }
}
```

- Para obter detalhes da API, consulte a [EnableRule](#) referência da API AWS SDK for Kotlin.

ListRuleNamesByTarget

O código de exemplo a seguir mostra como usar ListRuleNamesByTarget.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListRuleNamesByTarget](#) referência da API AWS SDK for Kotlin.

ListRules

O código de exemplo a seguir mostra como usar `ListRules`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
}
```

- Para obter detalhes da API, consulte a [ListRules](#) referência da API AWS SDK for Kotlin.

ListTargetsByRule

O código de exemplo a seguir mostra como usar `ListTargetsByRule`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListTargetsByRule](#) referência da API AWS SDK for Kotlin.

PutEvents

O código de exemplo a seguir mostra como usar PutEvents.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
```


- Para obter detalhes da API, consulte a [PutEvents](#) referência da API AWS SDK for Kotlin.

PutRule

O código de exemplo a seguir mostra como usar PutRule.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Criar uma regra agendada

```
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}")
    }
}
```

Crie uma regra que seja acionada quando um objeto é adicionado a um bucket do 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}")
    }
}
```

- Para obter detalhes da API, consulte a [PutRule](#) referência da API AWS SDK for Kotlin.

PutTargets

O código de exemplo a seguir mostra como usar PutTargets.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

Adicione um transformador de entrada a um destino para uma regra.

```
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)
    }
}
```

- Para obter detalhes da API, consulte a [PutTargets](#) referência da API AWS SDK for Kotlin.

RemoveTargets

O código de exemplo a seguir mostra como usar `RemoveTargets`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a [RemoveTargets](#) referência da API AWS SDK for Kotlin.

AWS Glue exemplos usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. AWS Glue

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um crawler que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados formatado em CSV.
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Criar um trabalho para extrair dados em CSV do bucket do S3, transformá-los e carregar a saída formatada em JSON em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte [Tutorial: Introdução ao AWS Glue Studio](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 targetObj =
    CrawlerTargets {
        s3Targets = targetList
    }

val crawlerRequest =
    CreateCrawlerRequest {
        databaseName = dbName
        name = crawlerName
        description = "Created by the AWS Glue Java API"
        targets = targetObj
        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")
    }
}
```


- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)
 - [GetJob](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)
 - [StartCrawler](#)
 - [StartJobRun](#)

Ações

CreateCrawler

O código de exemplo a seguir mostra como usar CreateCrawler.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")  
    }
```

```
}  
}
```

- Para obter detalhes da API, consulte a [CreateCrawler](#) referência da API AWS SDK for Kotlin.

GetCrawler

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")  
    }  
}
```

- Para obter detalhes da API, consulte a [GetCrawler](#) referência da API AWS SDK for Kotlin.

GetDatabase

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [GetDatabase](#) referência da API AWS SDK for Kotlin.

StartCrawler

O código de exemplo a seguir mostra como usar StartCrawler.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [StartCrawler](#) referência da API AWS SDK for Kotlin.

Exemplos de IAM usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

⚠ Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para Kotlin

i Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie as funções que envolvam ações do usuário do 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)
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

Ações

AttachRolePolicy

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
}
```

- Para obter detalhes da API, consulte a [AttachRolePolicy](#) referência da API AWS SDK for Kotlin.

CreateAccessKey

O código de exemplo a seguir mostra como usar CreateAccessKey.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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()
    }
}
```

- Para obter detalhes da API, consulte a [CreateAccessKey](#) referência da API AWS SDK for Kotlin.

CreateAccountAlias

O código de exemplo a seguir mostra como usar CreateAccountAlias.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [CreateAccountAlias](#) referência da API AWS SDK for Kotlin.

CreatePolicy

O código de exemplo a seguir mostra como usar CreatePolicy.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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()
    }
}
```

- Para obter detalhes da API, consulte a [CreatePolicy](#) referência da API AWS SDK for Kotlin.

CreateUser

O código de exemplo a seguir mostra como usar CreateUser.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [CreateUser](#) referência da API AWS SDK for Kotlin.

DeleteAccessKey

O código de exemplo a seguir mostra como usar DeleteAccessKey.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteAccessKey](#) referência da API AWS SDK for Kotlin.

DeleteAccountAlias

O código de exemplo a seguir mostra como usar DeleteAccountAlias.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteAccountAlias](#) referência da API AWS SDK for Kotlin.

DeletePolicy

O código de exemplo a seguir mostra como usar DeletePolicy.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }


    IAMClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- Para obter detalhes da API, consulte a [DeletePolicy](#) referência da API AWS SDK for Kotlin.

DeleteUser

O código de exemplo a seguir mostra como usar DeleteUser.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteUser](#) referência da API AWS SDK for Kotlin.

DetachRolePolicy

O código de exemplo a seguir mostra como usar DetachRolePolicy.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
}
```

- Para obter detalhes da API, consulte a [DetachRolePolicy](#) referência da API AWS SDK for Kotlin.

GetPolicy

O código de exemplo a seguir mostra como usar GetPolicy.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [GetPolicy](#) referência da API AWS SDK for Kotlin.

ListAccessKeys

O código de exemplo a seguir mostra como usar ListAccessKeys.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListAccessKeys](#) referência da API AWS SDK for Kotlin.

ListAccountAliases

O código de exemplo a seguir mostra como usar ListAccountAliases.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
suspend fun listAliases() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListAccountAliases](#) referência da API AWS SDK for Kotlin.

ListUsers

O código de exemplo a seguir mostra como usar ListUsers.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListUsers](#) referência da API AWS SDK for Kotlin.

UpdateUser

O código de exemplo a seguir mostra como usar UpdateUser.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [UpdateUser](#) referência da API AWS SDK for Kotlin.

AWS IoT exemplos usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. AWS IoT

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Olá AWS IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
        }
    }
}
```

```
    }  
  }  
}
```

- Para detalhes da API, consulte [listThings](#) em Referência de API AWS SDK para Kotlin.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como trabalhar com o gerenciamento de AWS IoT dispositivos.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 and 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")
    }
}
```

Ações

AttachThingPrincipal

O código de exemplo a seguir mostra como usar `AttachThingPrincipal`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [AttachThingPrincipal](#) referência da API AWS SDK for Kotlin.

CreateKeysAndCertificate

O código de exemplo a seguir mostra como usar CreateKeysAndCertificate.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [CreateKeysAndCertificate](#) referência da API AWS SDK for Kotlin.

CreateThing

O código de exemplo a seguir mostra como usar CreateThing.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal")
    }
}
```

```
}
```

- Para obter detalhes da API, consulte a [CreateThing](#) referência da API AWS SDK for Kotlin.

CreateTopicRule

O código de exemplo a seguir mostra como usar CreateTopicRule.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
        }
    }
```

- Para obter detalhes da API, consulte a [CreateTopicRule](#) referência da API AWS SDK for Kotlin.

DeleteCertificate

O código de exemplo a seguir mostra como usar DeleteCertificate.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteCertificate](#) referência da API AWS SDK for Kotlin.

DeleteThing

O código de exemplo a seguir mostra como usar DeleteThing.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteThing](#) referência da API AWS SDK for Kotlin.

DescribeEndpoint

O código de exemplo a seguir mostra como usar DescribeEndpoint.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
}
}

```

- Para obter detalhes da API, consulte a [DescribeEndpoint](#) referência da API AWS SDK for Kotlin.

DescribeThing

O código de exemplo a seguir mostra como usar DescribeThing.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
}

```


- Para obter detalhes da API, consulte a [DescribeThing](#) referência da API AWS SDK for Kotlin.

DetachThingPrincipal

O código de exemplo a seguir mostra como usar `DetachThingPrincipal`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DetachThingPrincipal](#) referência da API AWS SDK for Kotlin.

ListCertificates

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListCertificates](#) referência da API AWS SDK for Kotlin.

SearchIndex

O código de exemplo a seguir mostra como usar SearchIndex.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}") }
    }
}
```

- Para obter detalhes da API, consulte a [SearchIndex](#) referência da API AWS SDK for Kotlin.

UpdateThing

O código de exemplo a seguir mostra como usar UpdateThing.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
}
}
```

- Para obter detalhes da API, consulte a [UpdateThing](#) referência da API AWS SDK for Kotlin.

AWS IoT data exemplos usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. AWS IoT data

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos


- [Ações](#)

Ações

GetThingShadow

O código de exemplo a seguir mostra como usar GetThingShadow.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [GetThingShadow](#) referência da API AWS SDK for Kotlin.

UpdateThingShadow

O código de exemplo a seguir mostra como usar UpdateThingShadow.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [UpdateThingShadow](#) referência da API AWS SDK for Kotlin.

Exemplos do Amazon Keyspaces usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Keyspaces.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Olá, Amazon Keyspaces

Os exemplos de código a seguir mostram como começar a usar o Amazon Keyspaces.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListKeyspaces](#) referência da API AWS SDK for Kotlin.

Tópicos

- [Conceitos básicos](#)

- [Ações](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um keyspace e uma tabela. O esquema da tabela contém dados do filme e tem a point-in-time recuperação ativada.
- Conectar-se ao keyspace usando uma conexão TLS segura com autenticação SigV4.
- Consultar a tabela. Adicionar, recuperar e atualizar dados do filme.
- Atualizar a tabela. Adicionar uma coluna para rastrear os filmes assistidos.
- Restaurar a tabela ao estado anterior e limpar os recursos.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    collist.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}")
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)

- [ListKeyspaces](#)
- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

Ações

CreateKeyspace

O código de exemplo a seguir mostra como usar CreateKeyspace.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [CreateKeyspace](#) referência da API AWS SDK for Kotlin.

CreateTable

O código de exemplo a seguir mostra como usar CreateTable.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 collList = ArrayList<ColumnDefinition>()
    collList.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 = colList
    }

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}")
}
}
```

- Para obter detalhes da API, consulte a [CreateTable](#) referência da API AWS SDK for Kotlin.

DeleteKeyspace

O código de exemplo a seguir mostra como usar DeleteKeyspace.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- Para obter detalhes da API, consulte a [DeleteKeyspace](#) referência da API AWS SDK for Kotlin.

DeleteTable

O código de exemplo a seguir mostra como usar DeleteTable.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
```

- Para obter detalhes da API, consulte a [DeleteTable](#) referência da API AWS SDK for Kotlin.

GetKeyspace

O código de exemplo a seguir mostra como usar GetKeyspace.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [GetKeyspace](#) referência da API AWS SDK for Kotlin.

GetTable

O código de exemplo a seguir mostra como usar GetTable.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
        }
    }
}
```

```
    }  
  }  
}
```

- Para obter detalhes da API, consulte a [GetTable](#) referência da API AWS SDK for Kotlin.

ListKeyspaces

O código de exemplo a seguir mostra como usar `ListKeyspaces`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")  
            }  
    }  
}
```

- Para obter detalhes da API, consulte a [ListKeyspaces](#) referência da API AWS SDK for Kotlin.

ListTables

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
    }
}
```

- Para obter detalhes da API, consulte a [ListTables](#) referência da API AWS SDK for Kotlin.

RestoreTable

O código de exemplo a seguir mostra como usar RestoreTable.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}

```

- Para obter detalhes da API, consulte a [RestoreTable](#) referência da API AWS SDK for Kotlin.

UpdateTable

O código de exemplo a seguir mostra como usar UpdateTable.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
```

- Para obter detalhes da API, consulte a [UpdateTable](#) referência da API AWS SDK for Kotlin.

AWS KMS exemplos usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. AWS KMS

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos


- [Ações](#)

Ações

CreateAlias

O código de exemplo a seguir mostra como usar CreateAlias.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [CreateAlias](#) referência da API AWS SDK for Kotlin.

CreateGrant

O código de exemplo a seguir mostra como usar CreateGrant.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [CreateGrant](#) referência da API AWS SDK for Kotlin.

CreateKey

O código de exemplo a seguir mostra como usar CreateKey.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}

```

- Para obter detalhes da API, consulte a [CreateKey](#) referência da API AWS SDK for Kotlin.

Decrypt

O código de exemplo a seguir mostra como usar Decrypt.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
}

```

- Para obter detalhes da API, consulte [Decrypt](#) na Referência da API AWS SDK para Kotlin.

DescribeKey

O código de exemplo a seguir mostra como usar DescribeKey.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
}
}
```

- Para obter detalhes da API, consulte a [DescribeKey](#) referência da API AWS SDK for Kotlin.

DisableKey

O código de exemplo a seguir mostra como usar `DisableKey`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DisableKey](#) referência da API AWS SDK for Kotlin.

EnableKey

O código de exemplo a seguir mostra como usar `EnableKey`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [EnableKey](#) referência da API AWS SDK for Kotlin.

Encrypt

O código de exemplo a seguir mostra como usar Encrypt.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
```

- Para obter detalhes da API, consulte [Encrypt](#) na Referência da API AWS SDK para Kotlin.

ListAliases

O código de exemplo a seguir mostra como usar ListAliases.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListAliases](#) referência da API AWS SDK for Kotlin.

ListGrants

O código de exemplo a seguir mostra como usar ListGrants.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
}
```

- Para obter detalhes da API, consulte a [ListGrants](#) referência da API AWS SDK for Kotlin.

ListKeys

O código de exemplo a seguir mostra como usar ListKeys.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```


- Para obter detalhes da API, consulte a [ListKeys](#) referência da API AWS SDK for Kotlin.

Exemplos de Lambda usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)
- [Cenários](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte [Criar uma função do Lambda no console](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

Ações

CreateFunction

O código de exemplo a seguir mostra como usar CreateFunction.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [CreateFunction](#) referência da API AWS SDK for Kotlin.

DeleteFunction

O código de exemplo a seguir mostra como usar DeleteFunction.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteFunction](#) referência da API AWS SDK for Kotlin.

Invoke

O código de exemplo a seguir mostra como usar Invoke.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte [Invoke](#) na Referência da API AWS SDK para Kotlin.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

MediaConvert exemplos usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. MediaConvert

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos


- [Ações](#)

Ações

CreateJob

O código de exemplo a seguir mostra como usar CreateJob.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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> = mutableListOfOf()
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

```

```

H264ParControl.InitializeFromSource
H264QualityTuningLevel.SinglePass
{ qvbrQualityLevel = 8 }

H264FramerateControl.InitializeFromSource
    parControl =
    qualityTuningLevel =
    qvbrSettings = H264QvbrSettings
    codecLevel = H264CodecLevel.Auto
    codecProfile = H264CodecProfile.Main
    maxBitrate = 2400000
    framerateControl =
    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 =
    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 =

```

- Para obter detalhes da API, consulte a [CreateJob](#) referência da API AWS SDK for Kotlin.

GetJob

O código de exemplo a seguir mostra como usar GetJob.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}.")
}
```

- Para obter detalhes da API, consulte a [GetJob](#) referência da API AWS SDK for Kotlin.

ListJobs

O código de exemplo a seguir mostra como usar ListJobs.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListJobs](#) referência da API AWS SDK for Kotlin.

Exemplos do Amazon Pinpoint usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos


- [Ações](#)

Ações

CreateApp

O código de exemplo a seguir mostra como usar CreateApp.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestObj =
        CreateApplicationRequest {
            name = applicationName
        }


    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestObj
                },
            )
        return result.applicationResponse?.id
    }
}
```

- Para obter detalhes da API, consulte a [CreateApp](#) preferência da API AWS SDK for Kotlin.

CreateCampaign

O código de exemplo a seguir mostra como usar CreateCampaign.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [CreateCampaign](#) referência da API AWS SDK for Kotlin.

CreateSegment

O código de exemplo a seguir mostra como usar CreateSegment.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}

```

- Para obter detalhes da API, consulte a [CreateSegment](#) referência da API AWS SDK for Kotlin.

DeleteApp

O código de exemplo a seguir mostra como usar DeleteApp.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}

```

- Para obter detalhes da API, consulte a [DeleteApp](#) referência da API AWS SDK for Kotlin.

DeleteEndpoint

O código de exemplo a seguir mostra como usar DeleteEndpoint.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

suspend fun deletePinEndpoint(
    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")
    }
}

```

- Para obter detalhes da API, consulte a [DeleteEndpoint](#) referência da API AWS SDK for Kotlin.

GetEndpoint

O código de exemplo a seguir mostra como usar `GetEndpoint`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
```

- Para obter detalhes da API, consulte a [GetEndpoint](#) referência da API AWS SDK for Kotlin.

GetSegments

O código de exemplo a seguir mostra como usar GetSegments.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [GetSegments](#) referência da API AWS SDK for Kotlin.

SendMessage

O código de exemplo a seguir mostra como usar SendMessage.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 destinationOb =
        Destination {
            toAddresses = listOf(toAddressVal)
        }

    val emailContent =
        EmailContent {
            simple = message
        }
}
```

```
val sendEmailRequest =
    SendEmailRequest {
        fromEmailAddress = senderAddress
        destination = destination0b
        this.content = emailContent
    }

PinpointEmailClient { region = "us-east-1" }.use { pinpointemail ->
    pinpointemail.sendEmail(sendEmailRequest)
    println("Message Sent")
}
}
```

- Para obter detalhes da API, consulte a [SendMessages](#) referência da API AWS SDK for Kotlin.

Exemplos do Amazon RDS usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon RDS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)
- [Cenários](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de banco de dados e definir os valores dos parâmetros.
- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.
- Excluir a instância e o grupo de parâmetros.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 paraName: String
    if (dbParameters != null) {
        for (para in dbParameters) {
            // Only print out information about either auto_increment_offset or
            auto_increment_increment.
            paraName = para.parameterName.toString()
            if (paraName.compareTo("auto_increment_offset") == 0 ||
                paraName.compareTo("auto_increment_increment ") == 0) {
                println("*** The parameter name is $paraName")
                System.out.println("*** The parameter value is
                ${para.parameterValue}")
                System.out.println("*** The parameter data type is
                ${para.dataType}")
                System.out.println("*** The parameter description is
                ${para.description}")
                System.out.println("*** The parameter allowed values is
                ${para.allowedValues}")
            }
        }
    }
}

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 (engineOb in engines) {
                println("The name of the DB parameter group family for the database
                engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.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
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CriarDBInstance](#)
 - [Criar DBParameter grupo](#)
 - [CriarDBSnapshot](#)
 - [ExcluirDBInstance](#)
 - [Excluir DBParameter grupo](#)
 - [Descreva DBEngine as versões](#)
 - [DescreverDBInstances](#)
 - [Descreva DBParameter os grupos](#)
 - [DescreverDBParameters](#)
 - [DescreverDBSnapshots](#)
 - [DescribeOrderableDBInstanceOpções](#)
 - [Modificar DBParameter grupo](#)

Ações

CreateDBInstance

O código de exemplo a seguir mostra como usar CreateDBInstance.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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!")
}
```

- Para obter detalhes da API, consulte [Criar DBInstance](#) no AWS SDK para referência da API Kotlin.

DeleteDBInstance

O código de exemplo a seguir mostra como usar DeleteDBInstance.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte [Excluir DBInstance](#) no AWS SDK para referência da API Kotlin.

DescribeAccountAttributes

O código de exemplo a seguir mostra como usar DescribeAccountAttributes.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
}

```

- Para obter detalhes da API, consulte a [DescribeAccountAttributes](#) referência da API AWS SDK for Kotlin.

DescribeDBInstances

O código de exemplo a seguir mostra como usar DescribeDBInstances.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
}

```

- Para obter detalhes da API, consulte [Descrever DBInstances](#) no AWS SDK para referência da API Kotlin.

ModifyDBInstance

O código de exemplo a seguir mostra como usar `ModifyDBInstance`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte [Modificar DBInstance](#) no AWS SDK para referência da API Kotlin.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. [GitHub](#)

Serviços utilizados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos de serviços de dados do Amazon RDS usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon RDS Data Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Cenários](#)

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. [GitHub](#)

Serviços utilizados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon Redshift usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Redshift.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

CreateCluster

O código de exemplo a seguir mostra como usar CreateCluster.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie o 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}")
    }
}
```

- Para obter detalhes da API, consulte a [CreateCluster](#) referência da API AWS SDK for Kotlin.

DeleteCluster

O código de exemplo a seguir mostra como usar DeleteCluster.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Excluir o 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}")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteCluster](#) referência da API AWS SDK for Kotlin.

DescribeClusters

O código de exemplo a seguir mostra como usar DescribeClusters.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Descrever o 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}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a [DescribeClusters](#) referência da API AWS SDK for Kotlin.

ModifyCluster

O código de exemplo a seguir mostra como usar ModifyCluster.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Modificar um 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",
        )
    }
}
```

- Para obter detalhes da API, consulte a [ModifyCluster](#) referência da API AWS SDK for Kotlin.

Cenários

Criar uma aplicação Web para rastrear dados do Amazon Redshift

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia e relata itens de trabalho usando um banco de dados do Amazon Redshift.

SDK para Kotlin

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. [GitHub](#)

Serviços utilizados neste exemplo

- Amazon Redshift
- Amazon SES

Exemplos do Amazon Rekognition usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Rekognition.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

CompareFaces

O código de exemplo a seguir mostra como usar CompareFaces.

Para obter mais informações, consulte [Comparação de faces em imagens](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 uncompered = compareFacesResult.unmatchedFaces
    if (uncompered != null) {
        println("There was ${uncompered.size} face(s) that did not match")
    }

    println("Source image rotation:
${compareFacesResult.sourceImageOrientationCorrection}")
    println("target image rotation:
${compareFacesResult.targetImageOrientationCorrection}")
}
}

```

- Para obter detalhes da API, consulte a [CompareFaces](#) referência da API AWS SDK for Kotlin.

CreateCollection

O código de exemplo a seguir mostra como usar `CreateCollection`.

Para obter mais informações, consulte [Criar uma coleção](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```


- Para obter detalhes da API, consulte a [CreateCollection](#) referência da API AWS SDK for Kotlin.

DeleteCollection

O código de exemplo a seguir mostra como usar `DeleteCollection`.

Para obter mais informações, consulte [Excluir uma coleção](#).

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```


- Para obter detalhes da API, consulte a [DeleteCollection](#) referência da API AWS SDK for Kotlin.

DeleteFaces

O código de exemplo a seguir mostra como usar DeleteFaces.

Para obter mais informações, consulte [Excluir faces de uma coleção](#).

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DeleteFaces](#) referência da API AWS SDK for Kotlin.

DescribeCollection

O código de exemplo a seguir mostra como usar DescribeCollection.

Para obter mais informações, consulte [Descrever uma coleção](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

```
}  
}
```

- Para obter detalhes da API, consulte a [DescribeCollection](#) referência da API AWS SDK for Kotlin.

DetectFaces

O código de exemplo a seguir mostra como usar DetectFaces.

Para obter mais informações, consulte [Detectar faces em uma imagem](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")  
        }  
    }
```

```
}  
}
```

- Para obter detalhes da API, consulte a [DetectFaces](#) referência da API AWS SDK for Kotlin.

DetectLabels

O código de exemplo a seguir mostra como usar DetectLabels.

Para obter mais informações, consulte [Detectar rótulos em uma imagem](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")  
        }  
    }  
}
```

- Para obter detalhes da API, consulte a [DetectLabels](#) referência da API AWS SDK for Kotlin.

DetectModerationLabels

O código de exemplo a seguir mostra como usar DetectModerationLabels.

Para obter mais informações, consulte [Detectar imagens impróprias](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [DetectModerationLabels](#) referência da API AWS SDK for Kotlin.

DetectText

O código de exemplo a seguir mostra como usar DetectText.

Para obter mais informações, consulte [Detectar texto em uma imagem](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```


- Para obter detalhes da API, consulte a [DetectText](#) referência da API AWS SDK for Kotlin.

IndexFaces

O código de exemplo a seguir mostra como usar IndexFaces.

Para obter mais informações, consulte [Adicionar faces a uma coleção](#).

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
        }
    }
}
```

- Para obter detalhes da API, consulte a [IndexFaces](#) referência da API AWS SDK for Kotlin.

ListCollections

O código de exemplo a seguir mostra como usar `ListCollections`.

Para obter mais informações, consulte [Listar coleções](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListCollections](#) referência da API AWS SDK for Kotlin.

ListFaces

O código de exemplo a seguir mostra como usar ListFaces.

Para obter mais informações, consulte [Listar faces em uma coleção](#).

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```


- Para obter detalhes da API, consulte a [ListFaces](#) referência da API AWS SDK for Kotlin.

RecognizeCelebrities

O código de exemplo a seguir mostra como usar RecognizeCelebrities.

Para obter mais informações, consulte [Reconhecer celebridades em uma imagem](#).

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [RecognizeCelebrities](#) referência da API AWS SDK for Kotlin.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços utilizados neste exemplo


- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar informações em vídeos

O exemplo de código a seguir mostra como:

- Iniciar trabalhos do Amazon Rekognition para detectar elementos como pessoas, objetos e texto em vídeos.
- Verificar o status do trabalho até que os trabalhos terminem.
- Visualizar a lista de elementos detectados por cada trabalho.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Detecte faces em um vídeo armazenado em um bucket do Amazon S3.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vid0b =
        Video {
            s3object = s3obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vid0b
        }

    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}")
    }
}
}
```

Detecte conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket do Amazon S3.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vid0b =
        Video {
            s3object = s3obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vid0b
        }

    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)
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [GetCelebrityRecognition](#)
 - [GetContentModeration](#)
 - [GetLabelDetection](#)
 - [GetPersonTracking](#)
 - [GetSegmentDetection](#)
 - [GetTextDetection](#)
 - [StartCelebrityRecognition](#)
 - [StartContentModeration](#)
 - [StartLabelDetection](#)
 - [StartPersonTracking](#)
 - [StartSegmentDetection](#)

- [StartTextDetection](#)

Detectar objetos em imagens

O exemplo de código a seguir mostra como criar um aplicativo que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Exemplos de registro de domínios do Route 53 usando SDKs para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com registro de domínio do Route 53.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Registro de domínios do Olá, Route 53

O exemplo de código a seguir mostra como começar a usar o registro de domínios do Route 53.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListPrices](#) referência da API AWS SDK for Kotlin.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Listar os domínios atuais e as operações do ano passado.
- Ver o faturamento do ano passado e os preços dos tipos de domínio.
- Receber sugestões de domínio.
- Verificar a disponibilidade e a transferibilidade de um domínio.

- Opcionalmente, solicitar o registro de um domínio.
- Obter os detalhes de uma operação.
- Opcionalmente, obter os detalhes de um domínio.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CheckDomainAvailability](#)
 - [CheckDomainTransferability](#)
 - [GetDomainDetail](#)
 - [GetDomainSuggestions](#)
 - [GetOperationDetail](#)
 - [ListDomains](#)
 - [ListOperations](#)
 - [ListPrices](#)
 - [RegisterDomain](#)
 - [ViewBilling](#)

Ações

CheckDomainAvailability

O código de exemplo a seguir mostra como usar `CheckDomainAvailability`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [CheckDomainAvailability](#) referência da API AWS SDK for Kotlin.

CheckDomainTransferability

O código de exemplo a seguir mostra como usar CheckDomainTransferability.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [CheckDomainTransferability](#) referência da API AWS SDK for Kotlin.

GetDomainDetail

O código de exemplo a seguir mostra como usar GetDomainDetail.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [GetDomainDetail](#) referência da API AWS SDK for Kotlin.

GetDomainSuggestions

O código de exemplo a seguir mostra como usar `GetDomainSuggestions`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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(" ")
        }
    }
}
```

- Para obter detalhes da API, consulte a [GetDomainSuggestions](#) referência da API AWS SDK for Kotlin.

GetOperationDetail

O código de exemplo a seguir mostra como usar `GetOperationDetail`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [GetOperationDetail](#) referência da API AWS SDK for Kotlin.

ListDomains

O código de exemplo a seguir mostra como usar ListDomains.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}

```

- Para obter detalhes da API, consulte a [ListDomains](#) referência da API AWS SDK for Kotlin.

ListOperations

O código de exemplo a seguir mostra como usar ListOperations.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
            }
    }
}

```

```

        println("Date: ${content.submittedDate}")
    }
}
}

```

- Para obter detalhes da API, consulte a [ListOperations](#) referência da API AWS SDK for Kotlin.

ListPrices

O código de exemplo a seguir mostra como usar ListPrices.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
    }
}
}

```

- Para obter detalhes da API, consulte a [ListPrices](#) referência da API AWS SDK for Kotlin.

RegisterDomain

O código de exemplo a seguir mostra como usar RegisterDomain.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [RegisterDomain](#) referência da API AWS SDK for Kotlin.

ViewBilling

O código de exemplo a seguir mostra como usar ViewBilling.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
    }
}
```

- Para obter detalhes da API, consulte a [ViewBilling](#) referência da API AWS SDK for Kotlin.

Exemplos do Amazon S3 usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Conceitos básicos](#)
- [Ações](#)

- [Cenários](#)

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo para ele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Excluir os objetos do bucket e o bucket.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 copyBucketObj(
    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 delObj =
        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")
    }
}

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!")
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)

- [PutObject](#)

Ações

CopyObject

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
```

- Para obter detalhes da API, consulte a [CopyObject](#) referência da API AWS SDK for Kotlin.

CreateBucket

O código de exemplo a seguir mostra como usar CreateBucket.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [CreateBucket](#) referência da API AWS SDK for Kotlin.

CreateMultiRegionAccessPoint

O código de exemplo a seguir mostra como usar CreateMultiRegionAccessPoint.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Configure o cliente de controle do S3 para enviar a solicitação à região 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
}
```

Crie o ponto de acesso multirregional.

```
suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrappName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrappName
                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"
}

```

Aguarde até que o ponto de acesso multirregional fique disponível.

```

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)
    }
}

```



```
    }  
}
```

- Para ter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para Kotlin](#).
- Para obter detalhes da API, consulte a [CreateMultiRegionAccessPoint](#) referência da API AWS SDK for Kotlin.

DeleteBucketPolicy

O código de exemplo a seguir mostra como usar DeleteBucketPolicy.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).


```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- Para obter detalhes da API, consulte a [DeleteBucketPolicy](#) referência da API AWS SDK for Kotlin.

DeleteObjects

O código de exemplo a seguir mostra como usar DeleteObjects.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteObjects](#) referência da API AWS SDK for Kotlin.

GetBucketPolicy

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
    }
}
```

- Para obter detalhes da API, consulte a [GetBucketPolicy](#) referência da API AWS SDK for Kotlin.

GetObject

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
        }
    }
}
```

- Para obter detalhes da API, consulte a [GetObject](#) referência da API AWS SDK for Kotlin.

GetObjectAcl

O código de exemplo a seguir mostra como usar `GetObjectAcl`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}

```

- Para obter detalhes da API, consulte a [GetObjectAcl](#) referência da API AWS SDK for Kotlin.

ListObjectsV2

O código de exemplo a seguir mostra como usar ListObjectsV2.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
```

- Para obter detalhes da API, consulte [ListObjectsV2](#) no AWS SDK para referência da API Kotlin.

PutBucketAcl

O código de exemplo a seguir mostra como usar PutBucketAcl.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 = ownerObj
            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")
    }
}

```

- Para obter detalhes da API, consulte a [PutBucketAcl](#) referência da API AWS SDK for Kotlin.

PutObject

O código de exemplo a seguir mostra como usar PutObject.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [PutObject](#) referência da API AWS SDK for Kotlin.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie uma solicitação pré-assinada de `GetObject` e use o URL para fazer download de um objeto.

```
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
}

```

Crie uma solicitação `GetObject` atribuída previamente com opções avançadas.

```

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
}

```

Crie uma solicitação pré-assinada de `PutObject` e use-a para fazer upload de um objeto.

```
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.toByteArray())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- Para ter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para Kotlin](#).

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar objetos em imagens

O exemplo de código a seguir mostra como criar um aplicativo que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Obter um objeto de um ponto de acesso multirregional

O exemplo de código a seguir demonstra como obter um objeto de um ponto de acesso multirregional.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Configure o cliente do S3 para usar o algoritmo de assinatura Asymmetric 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
}
```

Use o ARN do ponto de acesso multirregional em vez de um nome de bucket para recuperar o objeto.

```
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 $mrpArn")
        }
    }
    return stringObj
}
```

- Para ter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para Kotlin](#).
- Para obter detalhes da API, consulte a [GetObject](#) referência da API AWS SDK for Kotlin.

SageMaker Exemplos de IA usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com IA. SageMaker

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Olá SageMaker AI

Os exemplos de código a seguir mostram como começar a usar a SageMaker IA.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListNotebookInstances](#) referência da API AWS SDK for Kotlin.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

CreatePipeline

O código de exemplo a seguir mostra como usar CreatePipeline.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
    }
}
```

- Para obter detalhes da API, consulte a [CreatePipeline](#) referência da API AWS SDK for Kotlin.

DeletePipeline

O código de exemplo a seguir mostra como usar DeletePipeline.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [DeletePipeline](#) referência da API AWS SDK for Kotlin.

DescribePipelineExecution

O código de exemplo a seguir mostra como usar DescribePipelineExecution.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
```

- Para obter detalhes da API, consulte a [DescribePipelineExecution](#) referência da API AWS SDK for Kotlin.

StartPipelineExecution

O código de exemplo a seguir mostra como usar `StartPipelineExecution`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
}
}
```

- Para obter detalhes da API, consulte a [StartPipelineExecution](#) referência da API AWS SDK for Kotlin.

Cenários


Conceitos básicos de trabalhos geoespaciais e pipelines

O exemplo de código a seguir mostra como:

- Configurar recursos para um pipeline.
- Configurar um pipeline que executa um trabalho geoespacial.
- Iniciar a execução de um pipeline.
- Monitorar o status da execução.
- Ver a saída do pipeline.
- Limpar recursos.

Para obter mais informações, consulte [Criar e executar SageMaker pipelines usando o AWS SDKs Community.aws](#).

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.waitForFunctionActive {
            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
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CreatePipeline](#)
 - [DeletePipeline](#)
 - [DescribePipelineExecution](#)
 - [StartPipelineExecution](#)
 - [UpdatePipeline](#)

Exemplos de Secrets Manager usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Ações](#)

Ações

GetSecretValue

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [GetSecretValue](#) referência da API AWS SDK for Kotlin.

Exemplos do Amazon SES usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon SES.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Cenários](#)

Cenários

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Criar uma aplicação Web para rastrear dados do Amazon Redshift

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia e relata itens de trabalho usando um banco de dados do Amazon Redshift.

SDK para Kotlin

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em [GitHub](#)

Serviços utilizados neste exemplo

- Amazon Redshift
- Amazon SES

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em [GitHub](#)

Serviços utilizados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Detectar objetos em imagens

O exemplo de código a seguir mostra como criar um aplicativo que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Exemplos do Amazon SNS usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Conceitos básicos

Olá, Amazon SNS

Os exemplos de código a seguir mostram como começar a usar o Amazon SNS.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListTopics](#) referência da API AWS SDK for Kotlin.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

CreateTopic

O código de exemplo a seguir mostra como usar CreateTopic.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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()
    }
}
```

- Para obter detalhes da API, consulte a [CreateTopic](#) referência da API AWS SDK for Kotlin.

DeleteTopic

O código de exemplo a seguir mostra como usar DeleteTopic.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteTopic](#) referência da API AWS SDK for Kotlin.

GetTopicAttributes

O código de exemplo a seguir mostra como usar GetTopicAttributes.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte a [GetTopicAttributes](#) referência da API AWS SDK for Kotlin.

ListSubscriptions

O código de exemplo a seguir mostra como usar ListSubscriptions.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListSubscriptions](#) referência da API AWS SDK for Kotlin.

ListTopics

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a [ListTopics](#) referência da API AWS SDK for Kotlin.

Publish

O código de exemplo a seguir mostra como usar `Publish`.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte [Publish](#) na Referência da API AWS SDK para Kotlin.

SetTopicAttributes

O código de exemplo a seguir mostra como usar SetTopicAttributes.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte a [SetTopicAttributes](#) referência da API AWS SDK for Kotlin.

Subscribe

O código de exemplo a seguir mostra como usar Subscribe.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Inscrever um endereço de e-mail em um tópico.

```
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()
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte [Subscribe](#) na Referência da API AWS SDK para Kotlin.

TagResource

O código de exemplo a seguir mostra como usar TagResource.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
    }
}
```

- Para obter detalhes da API, consulte a [TagResource](#) referência da API AWS SDK for Kotlin.

Unsubscribe

O código de exemplo a seguir mostra como usar Unsubscribe.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
    }
}
```

- Para obter detalhes da API, consulte [Cancelar assinatura](#) na Referência da API AWS SDK para Kotlin.

Cenários

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

O exemplo de código a seguir mostra como criar um aplicativo que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

SDK para Kotlin

Mostra como criar uma aplicação Android nativa que envia dados usando a API Kotlin do Amazon DynamoDB e envia uma mensagem de texto usando a API Kotlin do Amazon SNS.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Criação de uma aplicação do Amazon SNS

O exemplo de código a seguir mostra como criar um aplicativo que tenha funcionalidade de assinatura e publicação e traduza mensagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon SNS para criar uma aplicação com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e as instruções sobre como criar um aplicativo web, veja o exemplo completo em [GitHub](#).

Para ver o código-fonte completo e instruções sobre como criar um aplicativo Android nativo, veja o exemplo completo em [GitHub](#).

Serviços utilizados neste exemplo

- Amazon SNS
- Amazon Translate

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).


Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
    }
}
```

- Para obter detalhes da API, consulte [Publish](#) na Referência da API AWS SDK para Kotlin.

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquisar as filas para ver as mensagens recebidas.

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - [CreateQueue](#)
  - [CreateTopic](#)
  - [DeleteMessageBatch](#)
  - [DeleteQueue](#)
  - [DeleteTopic](#)
  - [GetQueueAttributes](#)
  - [Publicar](#)
  - [ReceiveMessage](#)
  - [SetQueueAttributes](#)
  - [Assinar](#)
  - [Cancelar assinatura](#)

# Exemplos do Amazon SQS usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

## Conceitos básicos

### Olá, Amazon SQS

Os exemplos de código a seguir mostram como começar a usar o Amazon SQS.

## SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
 }
 }
}
```

- Para obter detalhes da API, consulte a [ListQueues](#) referência da API AWS SDK for Kotlin.

## Tópicos

- [Ações](#)
- [Cenários](#)

## Ações

### CreateQueue

O código de exemplo a seguir mostra como usar CreateQueue.

#### SDK para Kotlin

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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()
}
}
```

- Para obter detalhes da API, consulte a [CreateQueue](#) referência da API AWS SDK for Kotlin.

## DeleteMessage

O código de exemplo a seguir mostra como usar DeleteMessage.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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!")
 }
}
```

- Para obter detalhes da API, consulte a [DeleteMessage](#) referência da API AWS SDK for Kotlin.

## DeleteQueue

O código de exemplo a seguir mostra como usar DeleteQueue.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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!")
 }
}
```

- Para obter detalhes da API, consulte a [DeleteQueue](#) referência da API AWS SDK for Kotlin.

## ListQueues

O código de exemplo a seguir mostra como usar ListQueues.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
 }
 }
}
```

```
}
```

- Para obter detalhes da API, consulte a [ListQueues](#) referência da API AWS SDK for Kotlin.

## ReceiveMessage

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
 }
 }
}
```

- Para obter detalhes da API, consulte a [ReceiveMessage](#) referência da API AWS SDK for Kotlin.

## SendMessage

O código de exemplo a seguir mostra como usar `SendMessage`.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
}
}
```

- Para obter detalhes da API, consulte a [SendMessage](#) referência da API AWS SDK for Kotlin.

## Cenários

### Crie um aplicativo de mensagem

O exemplo de código a seguir mostra como criar um aplicativo de mensagens usando o Amazon SQS.

#### SDK para Kotlin

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

#### Serviços utilizados neste exemplo

- Amazon Comprehend
- Amazon SQS

### Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.

- Publicar mensagens no tópico.
- Pesquisar as filas para ver as mensagens recebidas.

## SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - [CreateQueue](#)
  - [CreateTopic](#)
  - [DeleteMessageBatch](#)
  - [DeleteQueue](#)
  - [DeleteTopic](#)
  - [GetQueueAttributes](#)
  - [Publicar](#)
  - [ReceiveMessage](#)
  - [SetQueueAttributes](#)
  - [Assinar](#)
  - [Cancelar assinatura](#)

## Exemplos do Step Functions usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com Step Functions.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

## Conceitos básicos

### Olá, Step Functions

Os exemplos de código a seguir mostram como começar a usar o Step Functions.

## SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
 }
}
```

- Para obter detalhes da API, consulte a [ListStateMachines](#) referência da API AWS SDK for Kotlin.

## Tópicos

- [Conceitos básicos](#)
- [Ações](#)

## Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma atividade.
- Criar uma máquina de estado a partir de uma definição da Amazon States Language que contenha a atividade criada anteriormente como uma etapa.
- Executar a máquina de estado e responder à atividade com entrada do usuário.
- Obter o status e a saída finais após a conclusão da execução e, em seguida, limpar os recursos.

## SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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](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 = 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")
}

suspend fun sendTaskSuccess(
 token: String?,
 json: String?,
) {
 val successRequest =
 SendTaskSuccessRequest {
 taskToken = token
 output = json
 }
 SfnClient { region = "us-east-1" }.use { sfnClient ->
 sfnClient.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 { sfnClient ->
 val response = sfnClient.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
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - [CreateActivity](#)
  - [CreateStateMachine](#)
  - [DeleteActivity](#)
  - [DeleteStateMachine](#)
  - [DescribeExecution](#)
  - [DescribeStateMachine](#)
  - [GetActivityTask](#)
  - [ListActivities](#)

- [ListStateMachines](#)
- [SendTaskSuccess](#)
- [StartExecution](#)
- [StopExecution](#)

## Ações

### CreateActivity

O código de exemplo a seguir mostra como usar CreateActivity.

SDK para Kotlin

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```

- Para obter detalhes da API, consulte a [CreateActivity](#) referência da API AWS SDK for Kotlin.

### CreateStateMachine

O código de exemplo a seguir mostra como usar CreateStateMachine.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```


- Para obter detalhes da API, consulte a [CreateStateMachine](#) referência da API AWS SDK for Kotlin.

## DeleteActivity

O código de exemplo a seguir mostra como usar DeleteActivity.



## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
 }
}
```

- Para obter detalhes da API, consulte a [DeleteActivity](#) referência da API AWS SDK for Kotlin.

## DeleteStateMachine

O código de exemplo a seguir mostra como usar DeleteStateMachine.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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.")
}
}

```

- Para obter detalhes da API, consulte a [DeleteStateMachine](#) referência da API AWS SDK for Kotlin.

## DescribeExecution

O código de exemplo a seguir mostra como usar DescribeExecution.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
}
```

- Para obter detalhes da API, consulte a [DescribeExecution](#) referência da API AWS SDK for Kotlin.

## DescribeStateMachine

O código de exemplo a seguir mostra como usar DescribeStateMachine.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
 }
}
```

- Para obter detalhes da API, consulte a [DescribeStateMachinereferência](#) da API AWS SDK for Kotlin.

## GetActivityTask

O código de exemplo a seguir mostra como usar `GetActivityTask`.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```

- Para obter detalhes da API, consulte a [GetActivityTaskreferência](#) da API AWS SDK for Kotlin.

## ListActivities

O código de exemplo a seguir mostra como usar `ListActivities`.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
 }
 }
}
```

- Para obter detalhes da API, consulte a [ListActivities](#) referência da API AWS SDK for Kotlin.

## ListExecutions

O código de exemplo a seguir mostra como usar `ListExecutions`.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
 }
 }
 }
}

```

- Para obter detalhes da API, consulte a [ListExecutions](#) referência da API AWS SDK for Kotlin.

## ListStateMachines

O código de exemplo a seguir mostra como usar ListStateMachines.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
 }
 }
}
```

- Para obter detalhes da API, consulte a [ListStateMachines](#) referência da API AWS SDK for Kotlin.

## SendTaskSuccess

O código de exemplo a seguir mostra como usar SendTaskSuccess.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
}
}
```

- Para obter detalhes da API, consulte a [SendTaskSuccess](#) referência da API AWS SDK for Kotlin.

## StartExecution

O código de exemplo a seguir mostra como usar `StartExecution`.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```

- Para obter detalhes da API, consulte a [StartExecution](#) referência da API AWS SDK for Kotlin.



# Suporte exemplos usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. Suporte

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

## Conceitos básicos

### Olá Suporte

O exemplo de código a seguir mostra como começar a usar o Suporte.

### SDK para Kotlin

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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++
 }
 }
 }
}
```

- Para obter detalhes da API, consulte a [DescribeServices](#) referência da API AWS SDK for Kotlin.

## Tópicos

- [Conceitos básicos](#)
- [Ações](#)

## Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Obter e exibir os serviços disponíveis e os níveis de gravidade dos casos.
- Criar um caso de suporte usando um serviço, uma categoria e um nível de gravidade selecionados.
- Obter e exibir uma lista de casos em aberto para o dia atual.
- Adicionar um conjunto de anexos e uma comunicação ao novo caso.
- Descrever o novo anexo e a comunicação para o caso.
- Resolver o caso.
- Obter e exibir uma lista de casos resolvidos para o dia atual.

### SDK para Kotlin

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Ações

### AddAttachmentsToSet

O código de exemplo a seguir mostra como usar `AddAttachmentsToSet`.

SDK para Kotlin

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```

- Para obter detalhes da API, consulte a [AddAttachmentsToSet](#) referência da API AWS SDK for Kotlin.

### AddCommunicationToCase

O código de exemplo a seguir mostra como usar `AddCommunicationToCase`.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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")
 }
 }
}
```

- Para obter detalhes da API, consulte a [AddCommunicationToCase](#) referência da API AWS SDK for Kotlin.

## CreateCase

O código de exemplo a seguir mostra como usar CreateCase.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```

- Para obter detalhes da API, consulte a [CreateCase](#) referência da API AWS SDK for Kotlin.

## DescribeAttachment

O código de exemplo a seguir mostra como usar DescribeAttachment.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
 }
}
```

- Para obter detalhes da API, consulte a [DescribeAttachment](#) referência da API AWS SDK for Kotlin.

## DescribeCases

O código de exemplo a seguir mostra como usar DescribeCases.

## SDK para Kotlin

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
 }
}
}
```

- Para obter detalhes da API, consulte a [DescribeCases](#) referência da API AWS SDK for Kotlin.

## DescribeCommunications

O código de exemplo a seguir mostra como usar DescribeCommunications.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 ""
}

```

- Para obter detalhes da API, consulte a [DescribeCommunications](#) referência da API AWS SDK for Kotlin.

## DescribeServices

O código de exemplo a seguir mostra como usar DescribeServices.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
}
```

- Para obter detalhes da API, consulte a [DescribeServices](#) referência da API AWS SDK for Kotlin.

## DescribeSeverityLevels

O código de exemplo a seguir mostra como usar DescribeSeverityLevels.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
 }
}
```

- Para obter detalhes da API, consulte a [DescribeSeverityLevels](#) referência da API AWS SDK for Kotlin.

## ResolveCase

O código de exemplo a seguir mostra como usar `ResolveCase`.

SDK para Kotlin

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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}")
 }
}
```

- Para obter detalhes da API, consulte a [ResolveCase](#) referência da API AWS SDK for Kotlin.

## Exemplos do Amazon Translate usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Translate.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- [Cenários](#)

## Cenários

Criação de uma aplicação do Amazon SNS

O exemplo de código a seguir mostra como criar um aplicativo que tenha funcionalidade de assinatura e publicação e traduza mensagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon SNS para criar uma aplicação com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e as instruções sobre como criar um aplicativo web, veja o exemplo completo em [GitHub](#).

Para ver o código-fonte completo e instruções sobre como criar um aplicativo Android nativo, veja o exemplo completo em [GitHub](#).

## Serviços utilizados neste exemplo

- Amazon SNS
- Amazon Translate

# Segurança para o AWS SDK para Kotlin

A segurança da nuvem na Amazon Web Services (AWS) é a nossa maior prioridade. Como cliente da AWS, você contará com um data center e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança. A segurança é uma responsabilidade compartilhada entre você AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isso como a Segurança da nuvem e a Segurança na nuvem.

Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa todos os serviços oferecidos na AWS nuvem e fornecer serviços que você possa usar com segurança. Nossa responsabilidade de segurança é a maior prioridade em AWS, e a eficácia de nossa segurança é regularmente testada e verificada por auditores terceirizados como parte dos [Programas de AWS Conformidade](#).

Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você está usando e por outros fatores, incluindo a sensibilidade de seus dados, os requisitos da sua organização e as leis e regulamentos aplicáveis.

Esse AWS produto ou serviço segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do [AWS serviço, consulte a página de documentação de segurança](#) do serviço e os [AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

## Tópicos

- [Proteção de dados em AWS SDK para Kotlin](#)
- [AWS SDK para Kotlin suporte para TLS 1.2](#)
- [Gerenciamento de Identidade e Acesso](#)
- [Validação de conformidade para este AWS produto ou serviço](#)
- [Resiliência para este AWS produto ou serviço](#)
- [Segurança da infraestrutura para este AWS produto ou serviço](#)

## Proteção de dados em AWS SDK para Kotlin

O modelo de [responsabilidade AWS compartilhada modelo](#) se aplica à proteção de dados em AWS SDK para Kotlin. Conforme descrito neste modelo, AWS é responsável por proteger a

infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as [Data Privacy FAQ](#). Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog [AWS Shared Responsibility Model and RGPD](#) no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como [trabalhar com CloudTrail trilhas](#) no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para obter mais informações sobre os endpoints FIPS disponíveis, consulte [Federal Information Processing Standard \(FIPS\) 140-3](#).

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sigilosas, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o SDK para Kotlin ou outro Serviços da AWS usando o console, a API ou. AWS CLI AWS SDKs Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

# AWS SDK para Kotlin suporte para TLS 1.2

As informações a seguir se aplicam somente à implementação de Java SSL (a implementação SSL padrão na AWS SDK para Kotlin segmentação da JVM). Se você estiver usando uma implementação SSL diferente, consulte sua implementação SSL específica para saber como impor versões TLS.

## Suporte para TLS em Java

O TLS 1.2 tem suporte a partir do Java 7.

## Como verificar a versão do TLS

Para verificar qual versão do TLS é compatível com a sua máquina virtual Java (JVM), você pode usar o código a seguir.

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.toString(separator = ", "))
```

Para ver o handshake SSL em ação, e qual versão do TLS é usada, você pode usar a propriedade do sistema `javax.net.debug`.

```
-Djavax.net.debug=ssl
```

## Gerenciamento de Identidade e Acesso

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar AWS os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

### Tópicos

- [Público](#)
- [Autenticar com identidades](#)
- [Gerenciar o acesso usando políticas](#)
- [Como Serviços da AWS trabalhar com o IAM](#)

- [Solução de problemas AWS de identidade e acesso](#)

## Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz AWS.

**Usuário do serviço** — Se você Serviços da AWS costuma fazer seu trabalho, seu administrador fornece as credenciais e as permissões de que você precisa. À medida que você usa mais AWS recursos para fazer seu trabalho, talvez precise de permissões adicionais. Compreenda como o acesso é gerenciado pode ajudar a solicitar as permissões corretas ao administrador. Se você não conseguir acessar um recurso no AWS, consulte [Solução de problemas AWS de identidade e acesso](#) o guia do usuário do AWS service (Serviço da AWS) que você está usando.

**Administrador de serviços** — Se você é responsável pelos AWS recursos da sua empresa, provavelmente tem acesso total AWS a. É seu trabalho determinar quais AWS recursos e recursos seus usuários do serviço devem acessar. Envie as solicitações ao administrador do IAM para alterar as permissões dos usuários de serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM. Para saber mais sobre como sua empresa pode usar o IAM AWS, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

**Administrador do IAM:** se você for um administrador do IAM, talvez queira saber detalhes sobre como pode gravar políticas para gerenciar acesso ao AWS. Para ver exemplos de políticas AWS baseadas em identidade que você pode usar no IAM, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

## Autenticar com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.



Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte [Como fazer login Conta da AWS](#) no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para designar solicitações por conta própria, consulte [Versão 4 do AWS Signature para solicitações de API](#) no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser necessário fornecer informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte [Autenticação multifator](#) no Guia do usuário do AWS IAM Identity Center e [Usar a autenticação multifator da AWS no IAM](#) no Guia do usuário do IAM.

## Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário-raiz para tarefas diárias. Proteja as credenciais do usuário-raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário-raiz, consulte [Tarefas que exigem credenciais de usuário-raiz](#) no Guia do Usuário do IAM.

## Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade. Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, é recomendável usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas da AWS aplicativos. Para obter mais informações sobre o Centro de Identidade do IAM, consulte [O que é o Centro de Identidade do IAM?](#) no Guia do Usuário do AWS IAM Identity Center .

## Usuários e grupos do IAM

Um [usuário do IAM](#) é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, é recomendável contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, é recomendável alternar as chaves de acesso. Para obter mais informações, consulte [Alternar as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo](#) no Guia do Usuário do IAM.

Um [grupo do IAM](#) é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte [Casos de uso para usuários do IAM](#) no Guia do usuário do IAM.

## Perfis do IAM

Uma [função do IAM](#) é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Para assumir temporariamente uma função do IAM no AWS Management Console, você pode [alternar de um usuário para uma função do IAM \(console\)](#). Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para usar perfis, consulte [Métodos para assumir um perfil](#) no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- **Acesso de usuário federado:** para atribuir permissões a identidades federadas, é possível criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada,

essa identidade é associada ao perfil e recebe as permissões definidas por ele. Para ter mais informações sobre perfis para federação, consulte [Criar um perfil para um provedor de identidade de terceiros \(federação\)](#) no Guia do usuário do IAM. Se usar o Centro de Identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de Identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte [Conjuntos de Permissões](#) no Guia do Usuário do AWS IAM Identity Center .

- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões da entidade principal da chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado o principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte [Sessões de acesso direto](#).
- Perfil de serviço: um perfil de serviço é um [perfil do IAM](#) que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte [Criar um perfil para delegar permissões a um AWS service \(Serviço da AWS\)](#) no Guia do Usuário do IAM.
- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS) O serviço pode presumir o perfil de executar

uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

- Aplicativos em execução na Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo solicitações AWS CLI de AWS API. Isso é preferível a armazenar chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que os programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte [Usar uma função do IAM para conceder permissões a aplicativos executados em EC2 instâncias da Amazon](#) no Guia do usuário do IAM.

## Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte [Visão geral das políticas JSON](#) no Guia do usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e perfis não têm permissões. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

## Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte [Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente](#) no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte [Escolher entre políticas gerenciadas e políticas em linha](#) no Guia do usuário do IAM.

## Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. Você deve [especificar uma entidade principal](#) em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

## Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a [visão geral da lista de controle de acesso \(ACL\)](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

## Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- **Limites de permissões:** um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte [Limites de permissões para identidades do IAM](#) no Guia do usuário do IAM.
- **Políticas de controle de serviço (SCPs)** — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente várias Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations e SCPs, consulte [Políticas de controle de serviços](#) no Guia AWS Organizations do Usuário.
- **Políticas de controle de recursos (RCPs)** — RCPs são políticas JSON que você pode usar para definir o máximo de permissões disponíveis para recursos em suas contas sem atualizar as políticas do IAM anexadas a cada recurso que você possui. O RCP limita as permissões para recursos nas contas dos membros e pode afetar as permissões efetivas para identidades, incluindo a Usuário raiz da conta da AWS, independentemente de pertencerem à sua organização. Para obter mais informações sobre Organizations e RCPs, incluindo uma lista Serviços da AWS desse suporte RCPs, consulte [Políticas de controle de recursos \(RCPs\)](#) no Guia AWS Organizations do usuário.
- **Políticas de sessão:** são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recursos. Uma negação explícita em qualquer uma dessas políticas

substitui a permissão. Para obter mais informações, consulte [Políticas de sessão](#) no Guia do usuário do IAM.

## Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte [Lógica de avaliação de políticas](#) no Guia do usuário do IAM.

## Como Serviços da AWS trabalhar com o IAM

Para ter uma visão de alto nível de como Serviços da AWS funciona com a maioria dos recursos do IAM, consulte [AWS os serviços que funcionam com o IAM](#) no Guia do usuário do IAM.

Para saber como usar um específico AWS service (Serviço da AWS) com o IAM, consulte a seção de segurança do Guia do usuário do serviço relevante.

## Solução de problemas AWS de identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com AWS um IAM.

### Tópicos

- [Não estou autorizado a realizar uma ação em AWS](#)
- [Não estou autorizado a realizar iam: PassRole](#)
- [Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS recursos](#)

## Não estou autorizado a realizar uma ação em AWS

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo `my-example-widget` fictício, mas não tem as permissões `aws:GetWidget` fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário mateojackson deve ser atualizada para permitir o acesso ao recurso *my-example-widget* usando a ação *awes:GetWidget*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

## Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não está autorizado a executar a ação *iam:PassRole*, as suas políticas devem ser atualizadas para permitir que você passe uma função para o AWS.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazê-lo, você deve ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada *marymajor* tenta utilizar o console para executar uma ação no AWS. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação *iam:PassRole*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

## Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS recursos

É possível criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.



Para saber mais, consulte:

- Para saber se é AWS compatível com esses recursos, consulte [Como Serviços da AWS trabalhar com o IAM](#).
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como [fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui](#) no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como [fornecer acesso Contas da AWS a terceiros](#) no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte [Conceder acesso a usuários autenticados externamente \(federação de identidades\)](#) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

## Validação de conformidade para este AWS produto ou serviço

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte [Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS](#) e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de [AWS conformidade Programas AWS](#) de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte [Baixar relatórios em AWS Artifact](#) .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- [Governança e conformidade de segurança](#): esses guias de implementação de solução abordam considerações sobre a arquitetura e fornecem etapas para implantar recursos de segurança e conformidade.
- [Referência de serviços qualificados para HIPAA](#): lista os serviços qualificados para HIPAA. Nem todos Serviços da AWS são elegíveis para a HIPAA.
- AWS Recursos de <https://aws.amazon.com/compliance/resources/> de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.

- [AWS Guias de conformidade do cliente](#) — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).
- [Avaliação de recursos com regras](#) no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- [AWS Security Hub](#)— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a [Referência de controles do Security Hub](#).
- [Amazon GuardDuty](#) — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- [AWS Audit Manager](#)— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Esse AWS produto ou serviço segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do [AWS serviço, consulte a página de documentação de segurança](#) do serviço e os [AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

## Resiliência para este AWS produto ou serviço

A infraestrutura AWS global é construída em torno Regiões da AWS de zonas de disponibilidade.

Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância.

Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte [Infraestrutura AWS global](#).

Esse AWS produto ou serviço segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do [AWS serviço, consulte a página de documentação de segurança](#) do serviço e os [AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

## Segurança da infraestrutura para este AWS produto ou serviço

Esse AWS produto ou serviço usa serviços gerenciados e, portanto, é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte [AWS Cloud Security](#). Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte [Proteção](#) de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar este AWS Produto ou Serviço pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o [AWS Security Token Service](#) (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Esse AWS produto ou serviço segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do [AWS serviço, consulte a página de documentação de segurança](#) do serviço

---

e os [AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

# Histórico do documento

Este tópico descreve mudanças importantes no Guia do AWS SDK para Kotlin Desenvolvedor ao longo de sua história.

| Alteração                                                                          | Descrição                                                                                                                                                                  | Data                   |
|------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <a href="#">Proteção da integridade de dados com somas de verificação</a>          | Conteúdo atualizado com detalhes sobre o cálculo automático da soma de verificação.                                                                                        | 16 de janeiro de 2025  |
| <a href="#">Atualizar o conteúdo da cadeia de provedores de credenciais padrão</a> | <a href="#">A cadeia de fornecedores de credenciais padrão.</a>                                                                                                            | 15 de janeiro de 2025  |
| <a href="#">Exemplos de arquivos de compilação de atualização</a>                  | Mostre os elementos do arquivo de compilação conforme gerados pela versão 8.11.1 do Gradle. Mostre o uso do BOM. Incorpore links para a versão mais recente dos artefatos. | 18 de dezembro de 2024 |
| <a href="#">Adicionar tópico do DynamoDB Mapper (Developer Preview)</a>            | <a href="#">Mapeie classes para itens do DynamoDB usando o DynamoDB Mapper (Developer Preview)</a>                                                                         | 29 de outubro de 2024  |
| <a href="#">Atualize os nomes dos buckets do Amazon S3</a>                         | <a href="#">Somas de verificação do Amazon S3 com AWS SDK para Kotlin</a>                                                                                                  | 30 de setembro de 2024 |
| <a href="#">Adicione informações ao OkHttp 4 Engine</a>                            | <a href="#">Especifique um tipo de mecanismo HTTP</a>                                                                                                                      | 26 de setembro de 2024 |

|                                                                                                                             |                                                                                                     |                        |
|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|------------------------|
| <a href="#">Adicione informações sobre endpoints AWS baseados em contas para o DynamoDB</a>                                 | <a href="#">Use AWS endpoints baseados em conta</a>                                                 | 24 de setembro de 2024 |
| <a href="#">Adicionar um FAQs tópico de solução de problemas</a>                                                            | <a href="#">Solução de problemas FAQs</a>                                                           | 18 de setembro de 2024 |
| <a href="#">Exemplo OpenTelemetry de configuração de atualização e configuração do provedor de telemetria global padrão</a> | <a href="#">Observabilidade</a>                                                                     | 2 de maio de 2024      |
| <a href="#">Forneça mais detalhes sobre o processo de criação do cliente de serviço</a>                                     | <a href="#">Crie um cliente de serviço</a>                                                          | 14 de março de 2024    |
| <a href="#">Tópico Adicionar ponto de acesso multirregional</a>                                                             | <a href="#">Trabalhe com pontos de acesso multirregionais do Amazon S3 usando o SDK para Kotlin</a> | 6 de fevereiro de 2024 |
| <a href="#">Adicionar instruções do catálogo da versão do Gradle</a>                                                        | <a href="#">Catálogo de versões do Gradle (guia)</a>                                                | 19 de dezembro de 2023 |
| <a href="#">Versão de disponibilidade geral</a>                                                                             | <a href="#">AWS SDK para Kotlin Guia do desenvolvedor</a>                                           | 27 de novembro de 2023 |
| <a href="#">Atualize a seção de configuração dos endpoints do cliente com base nas atualizações do SDK</a>                  | <a href="#">Endpoints do cliente</a>                                                                | 25 de agosto de 2023   |
| <a href="#">Somos de verificação do Amazon S3</a>                                                                           | Seção adicionada sobre como usar somas de verificação flexíveis com o Amazon S3.                    | 14 de agosto de 2023   |
| <a href="#">Adicionar tópico de observabilidade</a>                                                                         | <a href="#">Observabilidade</a>                                                                     | 3 de agosto de 2023    |

|                                                                                                   |                                                                                                                                                   |                    |
|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| <a href="#">Adicione um tópico que discuta novas tentativas</a>                                   | <a href="#">Repetições</a>                                                                                                                        | 7 de julho de 2023 |
| <a href="#">Atualize a seção de configuração do cliente HTTP com base nas atualizações do SDK</a> | <a href="#">Configuração do cliente HTTP</a>                                                                                                      | 6 de junho de 2023 |
| <a href="#">Adicionar tópico de pré-assinatura HTTP</a>                                           | <a href="#">Solicitações pré-assinadas</a>                                                                                                        | 2 de junho de 2023 |
| <a href="#">Adicionar tópico sobre interceptores HTTP</a>                                         | <a href="#">Interceptores HTTP</a>                                                                                                                | 22 de maio de 2023 |
| <a href="#">Support para atualização automática de tokens</a>                                     | Atualize <a href="#">as instruções para acesso com login único.</a>                                                                               | 18 de maio de 2023 |
| <a href="#">Somos de verificação do Amazon S3</a>                                                 | Adicione uma seção que descreva como <a href="#">usar somas de verificação com o Amazon S3.</a>                                                   | 15 de maio de 2023 |
| <a href="#">Substituir a configuração do cliente de serviço</a>                                   | Adicione uma seção que descreva como <a href="#">substituir a configuração de um cliente de serviço e descreva como os recursos são afetados.</a> | 8 de maio de 2023  |
| <a href="#">Aplique uma versão mínima do TLS</a>                                                  | Adicione uma seção que descreva as opções para <a href="#">impor uma versão mínima do TLS.</a>                                                    | 3 de maio de 2023  |
| <a href="#">Configuração do endpoint do cliente</a>                                               | Adicione um tópico que discuta a configuração <a href="#">do endpoint do cliente.</a>                                                             | 7 de abril de 2023 |

---

|                                                                       |                                                                                                                                                                         |                       |
|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| <a href="#">Atualizações de práticas recomendadas do IAM</a>          | Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações, consulte <a href="#">Práticas recomendadas de segurança no IAM</a> . | 22 de março de 2023   |
| <a href="#">Adicionar um exemplo de arquivo de projeto Maven</a>      | Mostre um exemplo de um arquivo de projeto Maven, além de um arquivo de projeto no Gradle, no tópico <a href="#">Configurar</a> .                                       | 2 de dezembro de 2022 |
| <a href="#">Revise o conteúdo da versão prévia do Developer Guide</a> | Conteúdo atualizado para refletir o trabalho de desenvolvimento recente                                                                                                 | 04 de outubro de 2022 |
| <a href="#">AWS SDK para Kotlin Versão Developer Preview</a>          | <a href="#">AWS SDK para Kotlin</a>                                                                                                                                     | 2 de dezembro de 2021 |
| <a href="#">AWS SDK para Kotlin versão alfa</a>                       | <a href="#">Anunciando a nova AWS SDK para Kotlin versão alfa</a>                                                                                                       | 30 de agosto de 2021  |



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