Entwicklerhandbuch

AWS SDK für Kotlin

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SDK für Kotlin: Entwicklerhandbuch

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist der AWS SDK für Kotlin? 1
Fangen Sie mit dem SDK an 1
Wartung und Support für SDK-Hauptversionen 1
Weitere Ressourcen 1
Erste Schritte 3
Schritt 1: Einrichtung für dieses Tutorial 3
Schritt 2: Erstellen des Projekts 3
Schritt 3: Schreiben des Codes 6
Schritt 4: Anwendung erstellen und ausführen 8
Herzlichen Glückwunsch 9
Bereinigen 9
Nächste Schritte 9
Einrichten 10
Grundlegende Einrichtung 10
-Übersicht 10
Anmeldemöglichkeit beim AWS Zugriffsportal 12
Konfigurieren Sie Single Sign-On 12
Melden Sie sich mit dem AWS CLI 13
Installieren Sie Java und ein Build-Tool 14
Verwenden temporärer Anmeldeinformationen 14
Erstellen Sie Projekt-Build-Dateien 15
Codieren Sie Ihr Projekt 21
Loggen Sie sich mit dem ein AWS CLI 21
Konfiguration 22
Erstellen Sie einen Service-Client 25
Konfigurieren Sie einen Client im Code 25
Konfigurieren Sie einen Client aus der Umgebung 26
Schließen Sie den Client 27
AWS-Region Auswahl 28
Anbieterkette in der Standardregion 28
Anmeldeinformationsanbieter 28
Anbieterkette für Standardanmeldeinformationen 29
Geben Sie einen Anbieter für Anmeldeinformationen an 32
Client-Endpunkte 33

Benutzerdefinierte Konfiguration ... 34
Beispiele ... 37
HTTP ... 38
HTTP-Client-Konfiguration ... 38
Verwenden eines HTTP-Proxys ... 43
Interzeptoren ... 44
Erzwingen Sie eine TLS-Mindestversion ... 46
Erneute Versuche ... 47
Grundlegendes zum Verhalten bei Wiederholungen ... 47
Anpassen des Wiederholungsverhaltens ... 51
Beobachtbarkeit ... 60
Konfigurieren Sie ein `TelemetryProvider` ... 61
Kennzahlen ... 63
Protokollierung ... 65
Anbieter von Telemetriedaten ... 69
Überschreiben Sie die Client-Konfiguration ... 70
Lebenszyklus eines Kunden außer Kraft gesetzt ... 71
Gemeinsam genutzte -Ressourcen ... 72
Verwenden Sie das SDK ... 73
Anfragen stellen ... 73
DSL-Überladungen der Serviceschnittstelle ... 74
Anfragen ohne erforderliche Eingaben ... 75
Coroutinen ... 75
Gleichzeitige Anfragen stellen ... 75
Blockierungsanfragen stellen ... 77
Streaming-Operationen ... 77
Streaming-Antworten ... 77
Streaming-Anfragen ... 78
Paginierung ... 79
Waiter ... 80
Fehlerbehandlung ... 81
Ausnahmen für Dienste ... 82
Ausnahmen für Kunden ... 82
Fehler-Metadaten ... 82
Anfragen vorab signieren ... 83
Grundlagen der Vorsignierung ... 83

Erweiterte Presigning-Konfiguration ... 84
Vorsignierung von POST- und PUT-Anfragen ... 84
Operationen, die das SDK vorab signieren kann ... 86
Problembehebung FAQs ... 86
Wie behebe ich Probleme mit dem Hinweis „Verbindung geschlossen“? ... 86
Warum werden Ausnahmen ausgelöst, bevor die maximale Anzahl an Versuchen erreicht ist? ... 87
Wie behebe `NoSuchMethodError` ich oder? NoClassDefFoundError ... 89
Wie löse ich Abhängigkeitskonflikte? ... 89
Spöttisch ... 91
MockK ... 91
Arbeite mit AWS-Services ... 97
Amazon S3 ... 98
Schutz der Datenintegrität mit Prüfsummen ... 99
Arbeiten Sie mit Access Points für mehrere Regionen ... 104
DynamoDB ... 109
Verwenden Sie kontobasierte Endpunkte AWS ... 109
DynamoDB Mapper verwenden (Developer Preview) ... 110
Codebeispiele ... 138
API Gateway ... 139
Szenarien ... 140
Aurora ... 140
Grundlagen ... 141
Aktionen ... 154
Szenarien ... 140
Auto Scaling ... 169
Grundlagen ... 141
Aktionen ... 154
Amazon Bedrock ... 186
Aktionen ... 154
Amazon Bedrock Runtime ... 187
Amazon Nova ... 188
CloudWatch ... 192
Erste Schritte ... 192
Grundlagen ... 141
Aktionen ... 154

CloudWatch Logs 232
Aktionen 154
Amazon Cognito Identity Provider 236
Aktionen 154
Szenarien 140
Amazon Comprehend 252
Szenarien 140
DynamoDB 252
Grundlagen 141
Aktionen 154
Szenarien 140
Amazon EC2 281
Erste Schritte 192
Grundlagen 141
Aktionen 154
Amazon ECR 311
Erste Schritte 192
Grundlagen 141
Aktionen 154
OpenSearch Dienst 341
Aktionen 154
EventBridge 345
Erste Schritte 192
Grundlagen 141
Aktionen 154
AWS Glue 376
Grundlagen 141
Aktionen 154
IAM 387
Grundlagen 141
Aktionen 154
AWS IoT 407
Erste Schritte 192
Grundlagen 141
Aktionen 154
AWS IoT data 430

Aktionen ... 154
AWS IoT FleetWise ... 433
Erste Schritte ... 192
Grundlagen ... 141
Aktionen ... 154
Amazon Keyspaces ... 468
Erste Schritte ... 192
Grundlagen ... 141
Aktionen ... 154
AWS KMS ... 493
Aktionen ... 154
Lambda ... 502
Grundlagen ... 141
Aktionen ... 154
Szenarien ... 140
Amazon Location ... 512
Erste Schritte ... 192
Grundlagen ... 141
Aktionen ... 154
MediaConvert ... 543
Aktionen ... 154
Amazon Pinpoint ... 548
Aktionen ... 154
Amazon RDS ... 558
Grundlagen ... 141
Aktionen ... 154
Szenarien ... 140
Amazon RDS Data Service ... 576
Szenarien ... 140
Amazon Redshift ... 577
Aktionen ... 154
Szenarien ... 140
Amazon Rekognition ... 581
Aktionen ... 154
Szenarien ... 140
Route-53-Domainregistrierung ... 599

Erste Schritte ... 192
Grundlagen ... 141
Aktionen ... 154
Amazon S3 ... 619
Grundlagen ... 141
Aktionen ... 154
Szenarien ... 140
SageMaker KI ... 641
Erste Schritte ... 192
Aktionen ... 154
Szenarien ... 140
Secrets Manager ... 666
Aktionen ... 154
Amazon SES ... 667
Szenarien ... 140
Amazon SNS ... 669
Erste Schritte ... 192
Aktionen ... 154
Szenarien ... 140
Amazon SQS ... 696
Erste Schritte ... 192
Aktionen ... 154
Szenarien ... 140
Step Functions ... 719
Erste Schritte ... 192
Grundlagen ... 141
Aktionen ... 154
Support ... 740
Erste Schritte ... 192
Grundlagen ... 141
Aktionen ... 154
Amazon Translate ... 758
Szenarien ... 140
Sicherheit ... 760
Datenschutz ... 761
Erzwingen von TLS 1.2 ... 762

TLS-Unterstützung in Java 762
Vorgehensweise zum Überprüfen der TLS-Version 762
Identitäts- und Zugriffsverwaltung 762
Zielgruppe 763
Authentifizierung mit Identitäten 763
Verwalten des Zugriffs mit Richtlinien 765
Wie AWS-Services arbeiten Sie mit IAM 767
Fehlerbehebung bei AWS Identität und Zugriff 767
Compliance-Validierung 769
Ausfallsicherheit 770
Infrastruktursicherheit 770
Dokumentverlauf 772
..... dcclxxvii

Was ist der AWS SDK für Kotlin?

Das AWS SDK für Kotlin stellt Kotlin-APIs für Amazon Web Services bereit. Mit dem SDK können Sie Kotlin-Anwendungen erstellen, die mit Amazon S3, Amazon EC2, Amazon DynamoDB und mehr funktionieren. Mit dem Kotlin-SDK können Sie auf die JVM-Plattform oder die Android-API Level 24 oder höher abzielen. Support für weitere Plattformen wie JavaScript und Native wird in future Versionen verfügbar sein.

Informationen zu den kommenden Funktionen in future Versionen finden Sie in unserer Roadmap unter GitHub.

Fangen Sie mit dem SDK an

Folgen Sie dem Erste Schritte Tutorial, um mit dem SDK zu beginnen.

Informationen zum Einrichten Ihrer Entwicklungsumgebung finden Sie unterEinrichten.

Informationen zum Erstellen und Konfigurieren von Service-Clients für Anfragen finden Sie unter Konfiguration. AWS-Services Informationen zu den verschiedenen Funktionen des SDK finden Sie unterVerwenden Sie das SDK.

Anwendungsfälle und Beispiele für die Ausführung bestimmter API-Operationen finden Sie unterCodebeispiele.

Wartung und Support für SDK-Hauptversionen

Informationen zur Wartung und zum Support für SDK-Hauptversionen und die ihnen zugrunde liegenden Abhängigkeiten finden Sie in den folgenden Themen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch:

- AWS SDKs und Richtlinien zur Wartung von Tools
- AWS SDKs Matrix zur Support von Versionen und Tools

Weitere Ressourcen

Zusätzlich zu diesem Handbuch finden Sie im Folgenden wertvolle Online-Ressourcen für SDK für Kotlin-Entwickler:

- AWS Entwickler-Blog
- Foren für Entwickler
- SDK-Quelle (GitHub)
- AWS-Codebeispiel-Katalog
- @awsdevelopers (X, früher Twitter)

Fangen Sie mit dem SDK für Kotlin an

Das AWS SDK für Kotlin bietet Kotlin APIs für jeden AWS-Service. Mit dem SDK können Sie Kotlin-Anwendungen erstellen, die mit Amazon S3, Amazon EC2, Amazon DynamoDB und mehr funktionieren.

Dieses Tutorial zeigt Ihnen, wie Sie Gradle verwenden, um Abhängigkeiten für zu definieren. AWS SDK für Kotlin Anschließend erstellen Sie Code, der Daten in eine DynamoDB-Tabelle schreibt. Obwohl Sie vielleicht die Funktionen einer IDE verwenden möchten, benötigen Sie für dieses Tutorial lediglich ein Terminalfenster und einen Texteditor.

Gehen Sie wie folgt vor, um dieses Tutorial abzuschließen:

- Schritt 1: Richten Sie sich für dieses Tutorial ein
- Schritt 2: Erstellen Sie das Projekt
- Schritt 3: Schreiben Sie den Code
- Schritt 4: Erstellen Sie die Anwendung und führen Sie sie aus

Schritt 1: Einrichtung für dieses Tutorial

Bevor Sie mit diesem Tutorial beginnen, benötigen Sie einen IAM Identity Center-Berechtigungssatz, der auf DynamoDB zugreifen kann, und Sie benötigen eine Kotlin-Entwicklungsumgebung, die mit IAM Identity Center Single Sign-On-Einstellungen konfiguriert ist, auf die Sie zugreifen können. AWS

Folgen Sie den Anweisungen in diesem Handbuch, um die Grundlegende Einrichtung Grundlagen für dieses Tutorial einzurichten.

Nachdem Sie Ihre Entwicklungsumgebung mit Single Sign-On-Zugriff für das Kotlin SDK konfiguriert haben und Sie eine aktive AWS Access-Portal-Sitzung haben, fahren Sie mit Schritt 2 fort.

Schritt 2: Erstellen des Projekts

Um das Projekt für dieses Tutorial zu erstellen, verwenden Sie zunächst Gradle, um die Basisdateien für ein Kotlin-Projekt zu erstellen. Aktualisieren Sie dann die Dateien mit den erforderlichen Einstellungen, Abhängigkeiten und dem Code für. AWS SDK für Kotlin

Um ein neues Projekt mit Gradle zu erstellen

> **Note**
>
> Dieses Tutorial verwendet Gradle Version 8.11.1 mit dem `gradle init` Befehl, der in Schritt 3 unten fünf Eingabeaufforderungen bietet. Wenn Sie eine andere Version von Gradle verwenden, können sich die Eingabeaufforderungen unterscheiden, ebenso wie die vorausgefüllten Versionen von Artefakten.

1. Erstellen Sie ein neues Verzeichnis mit einem Namen `getstarted` an einem Ort Ihrer Wahl, z. B. Ihrem Desktop oder Home-Ordner.
2. Öffnen Sie ein Terminal- oder Befehlszeilenfenster und navigieren Sie zu dem `getstarted` Verzeichnis, das Sie erstellt haben.
3. Verwenden Sie den folgenden Befehl, um ein neues Gradle-Projekt und eine grundlegende Kotlin-Klasse zu erstellen.

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

   - Wenn Sie nach dem Ziel gefragt werden`Java version`, drücken Sie `Enter` (Standardeinstellung). `21`
   - Wenn Sie dazu aufgefordert werden`Project name`, drücken Sie `Enter` (`getstarted`in diesem Tutorial wird standardmäßig der Verzeichnisname verwendet).
   - Wenn Sie dazu aufgefordert werden`application structure`, drücken Sie `Enter` (standardmäßig). `Single application project`
   - Wenn Sie dazu aufgefordert werden`Select test framework`, drücken Sie `Enter` (standardmäßig). `kotlin.test`
   - Wenn Sie dazu aufgefordert werden`Generate build using new APIs and behavior`, drücken Sie `Enter` (standardmäßig). `no`

Um Ihr Projekt mit Abhängigkeiten für Amazon S3 AWS SDK für Kotlin und Amazon S3 zu konfigurieren

- Ersetzen Sie in dem `getstarted` Verzeichnis, das Sie im vorherigen Verfahren erstellt haben, den Inhalt der `settings.gradle.kts` Datei durch den folgenden Inhalt und *`X.Y.Z`* ersetzen Sie ihn durch die neueste Version des SDK für 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")
```

- Navigieren Sie zu dem `gradle` Verzeichnis innerhalb des `getstarted` Verzeichnisses. Ersetzen Sie den Inhalt der benannten Versionskatalogdatei `libs.versions.toml` durch den folgenden Inhalt:

```
[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" }
```

- Navigieren Sie zu dem Verzeichnis `app` und öffnen Sie die Datei `build.gradle.kts`. Ersetzen Sie seinen Inhalt durch den folgenden Code und speichern Sie dann Ihre Änderungen.

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


dependencies {
```

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

Der `dependencies` Abschnitt enthält einen `implementation` Eintrag für das Amazon S3 S3-Modul von AWS SDK für Kotlin. Der Gradle-Compiler ist in diesem Abschnitt für die Verwendung von Java 2.1 konfiguriert. `java`

Schritt 3: Schreiben des Codes

Nachdem das Projekt erstellt und konfiguriert wurde, bearbeiten Sie die Standardklasse des Projekts, App um den folgenden Beispielcode zu verwenden.

1. Navigieren Sie in Ihrem Projektordner `app` zum Verzeichnis`src/main/kotlin/org/example`. Öffnen Sie die `App.kt` Datei.
2. Ersetzen Sie den Inhalt durch den folgenden Code und speichern Sie die Datei.

```
package org.example

import aws.sdk.kotlin.services.s3.*
import aws.sdk.kotlin.services.s3.model.BucketLocationConstraint
import aws.smithy.kotlin.runtime.content.ByteStream
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!")
}
```

Schritt 4: Anwendung erstellen und ausführen

Nachdem das Projekt erstellt wurde und die Beispielklasse enthält, erstellen Sie die Anwendung und führen Sie sie aus.

1. Öffnen Sie ein Terminal- oder ein Befehlszeilenfenster und navigieren Sie zu Ihrem Projektverzeichnis `getstarted`.
2. Verwenden Sie den folgenden Befehl, um Ihre Anwendung zu erstellen und auszuführen:

```
gradle run
```

> **Note**
>
> Wenn Sie eine erhaltenIdentityProviderException, haben Sie möglicherweise keine aktive Single Sign-On-Sitzung. Führen Sie den `aws sso login` AWS CLI-Befehl aus, um eine neue Sitzung zu initiieren.

Die Anwendung ruft den API-Vorgang CreateBucket auf, um einen neuen S3-Bucket zu erstellen, und ruft dann PutObject auf, um ein neues Objekt in den neuen S3-Bucket einzufügen.

In der `cleanUp()` Funktion am Ende löscht die Anwendung das Objekt und anschließend den S3-Bucket.

Um die Ergebnisse in der Amazon S3 S3-Konsole zu sehen

1. Kommentieren Sie die Zeile `cleanUp(s3)` im `runBlocking` Abschnitt aus und speichern Sie die Datei. `App.kt`

2. Erstellen Sie das Projekt neu und fügen Sie ein neues Objekt in einen neuen S3-Bucket ein, indem Sie Folgendes ausführen`gradle run`.
3. Melden Sie sich bei der Amazon S3 S3-Konsole an, um das neue Objekt im neuen S3-Bucket anzusehen.

Nachdem Sie das Objekt angesehen haben, löschen Sie den S3-Bucket.

Herzlichen Glückwunsch

Wenn Ihr Gradle-Projekt ohne Fehler erstellt und ausgeführt wurde, dann herzlichen Glückwunsch. Sie haben erfolgreich Ihre erste Kotlin-Anwendung mit dem erstellt. AWS SDK für Kotlin

Bereinigen

Wenn Sie mit der Entwicklung Ihrer neuen Anwendung fertig sind, löschen Sie alle AWS Ressourcen, die Sie in diesem Tutorial erstellt haben, um Gebühren zu vermeiden. Möglicherweise möchten Sie auch den Projektordner (`get-started`), den Sie in Schritt 2 erstellt haben, löschen oder archivieren.

Gehen Sie wie folgt vor, um Ressourcen zu bereinigen:

- Wenn Sie den `cleanUp()` Funktionsaufruf auskommentiert haben, löschen Sie den S3-Bucket mithilfe der Amazon S3 S3-Konsole.

Nächste Schritte

Nachdem Sie sich mit den Grundlagen vertraut gemacht haben, können Sie sich über Folgendes informieren:

- Zusätzliche Einrichtungsschritte für die Arbeit mit dem SDK für Kotlin
- Konfiguration des SDK für Kotlin
- Verwenden des SDK für Kotlin
- Sicherheit für das SDK für Kotlin

Richten Sie das ein AWS SDK für Kotlin

Um Anfragen zur AWS-Services Verwendung von zu stellen AWS SDK für Kotlin, benötigen Sie Folgendes:

- Die Möglichkeit, sich beim AWS Zugangsportal anzumelden
- Erlaubnis zur Nutzung der AWS Ressourcen, die Ihre Anwendung benötigt
- Eine Entwicklungsumgebung mit den folgenden Elementen:
  - Gemeinsam genutzte Konfigurationsdateien, die auf mindestens eine der folgenden Arten eingerichtet wurden:
    - Die `config` Datei enthält Einstellungen für die IAM Identity Center-Anmeldeinformationen, sodass das SDK Anmeldeinformationen abrufen AWS kann
    - Die `credentials` Datei enthält temporäre Anmeldeinformationen
  - Ein Tool zur Build-Automatisierung wie Gradle oder Maven
- Eine aktive AWS Access-Portal-Sitzung, wenn Sie bereit sind, Ihre Anwendung auszuführen

In diesem Thema

- Grundlegende Einrichtung
- Erstellen Sie Projekt-Build-Dateien
- Codieren Sie Ihr Kotlin-Projekt mit dem SDK für Kotlin

Grundlegende Einrichtung

-Übersicht

Um erfolgreich Anwendungen entwickeln zu können, die AWS-Services über das zugreifen AWS SDK für Kotlin, müssen die folgenden Anforderungen erfüllt sein.

- Sie müssen sich bei dem AWS Zugriffsportal anmelden können, das im verfügbar ist AWS IAM Identity Center.
- Die für das SDK konfigurierten Berechtigungen der IAM-Rolle müssen den Zugriff auf die AWS-Services , die Ihre Anwendung benötigt, ermöglichen. Die mit der PowerUserAccess AWS verwalteten Richtlinie verbundenen Berechtigungen reichen für die meisten Entwicklungsanforderungen aus.

- Eine Entwicklungsumgebung mit den folgenden Elementen:
  - Gemeinsam genutzte Konfigurationsdateien, die auf mindestens eine der folgenden Arten eingerichtet wurden:
    - Die `config` Datei enthält Single-Sign-On-Einstellungen für IAM Identity Center, sodass das SDK Anmeldeinformationen abrufen AWS kann.
    - Die `credentials` Datei enthält temporäre Anmeldeinformationen.
  - Eine Installation von Java 8 oder höher.
  - Ein Tool zur Build-Automatisierung wie Maven oder Gradle.
  - Ein Texteditor für die Arbeit mit Code.
  - (Optional, aber empfohlen) Eine IDE (integrierte Entwicklungsumgebung) wie IntelliJ IDEA oder Eclipse.

    Wenn Sie eine IDE verwenden, können Sie AWS Toolkit s auch integrieren, um einfacher damit zu arbeiten. AWS-Services Die AWS Toolkit for IntelliJund AWS Toolkit for Eclipsesind zwei Toolkits, die Sie verwenden können.

- Eine aktive AWS Access-Portal-Sitzung, wenn Sie bereit sind, Ihre Anwendung auszuführen. Sie verwenden den AWS Command Line Interface , um den Anmeldevorgang für das AWS Zugangsportal von IAM Identity Center zu initiieren.

> **Important**
>
> Bei den Anweisungen in diesem Abschnitt zur Einrichtung wird davon ausgegangen, dass Sie oder Ihre Organisation IAM Identity Center verwenden. Wenn Ihre Organisation einen externen Identitätsanbieter verwendet, der unabhängig von IAM Identity Center arbeitet, finden Sie heraus, wie Sie temporäre Anmeldeinformationen für das SDK für Kotlin erhalten können. Folgen Sie diesen Anweisungen, um der Datei temporäre Anmeldeinformationen hinzuzufügen. `~/.aws/credentials`
> Wenn Ihr Identitätsanbieter der `~/.aws/credentials` Datei automatisch temporäre Anmeldeinformationen hinzufügt, stellen Sie sicher, dass der Profilname `[default]` so lautet, dass Sie dem SDK keinen Profilnamen angeben müssen oder AWS CLI.

Anmeldemöglichkeit beim AWS Zugriffsportal

Das AWS Zugriffsportal ist die Webadresse, über die Sie sich manuell beim IAM Identity Center anmelden. Das Format der URL ist `d-xxxxxxxxxx.awsapps.com/start` oder*your_subdomain*`.awsapps.com/start`.

Wenn Sie mit dem AWS Zugriffsportal nicht vertraut sind, folgen Sie den Anleitungen für den Kontozugriff im Thema IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Richten Sie den Single Sign-On-Zugriff für das SDK ein

Nachdem Sie Schritt 2 im Abschnitt Programmatischer Zugriff abgeschlossen haben, damit das SDK die IAM Identity Center-Authentifizierung verwenden kann, sollte Ihr System die folgenden Elemente enthalten.

- Die AWS CLI, mit der Sie eine AWS Access-Portal-Sitzung starten, bevor Sie Ihre Anwendung ausführen.
- Eine `~/.aws/config` Datei, die ein Standardprofil enthält. Das SDK für Kotlin verwendet die SSO-Token-Provider-Konfiguration des Profils, um Anmeldeinformationen abzurufen, bevor Anfragen an AWS gesendet werden. Der Wert `sso_role_name`, bei dem es sich um eine IAM-Rolle handelt, die mit einem Berechtigungssatz von IAM Identity Center verbunden ist, sollte den Zugriff auf die in Ihrer Anwendung verwendeten AWS-Services ermöglichen.

  Die folgende `config` Beispieldatei zeigt ein Standardprofil, das mit der Konfiguration des SSO-Token-Anbieters eingerichtet wurde. Die `sso_session`-Einstellung des Profils bezieht sich auf den benannten `sso-session`-Abschnitt. Der `sso-session` Abschnitt enthält Einstellungen zum Initiieren einer AWS Access-Portal-Sitzung.

```
[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
```

Weitere Informationen zu den in der Konfiguration des SSO-Token-Anbieters verwendeten Einstellungen finden Sie unter Konfiguration des SSO-Token-Anbieters im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Wenn Ihre Entwicklungsumgebung nicht wie zuvor gezeigt für den programmatischen Zugriff eingerichtet ist, folgen Sie Schritt 2 im SDKs Referenzhandbuch.

Melden Sie sich mit dem AWS CLI

Bevor Sie eine Zugriffsanwendung ausführen AWS-Services, benötigen Sie eine aktive AWS Access-Portal-Sitzung, damit das SDK die IAM Identity Center-Authentifizierung zur Auflösung von Anmeldeinformationen verwenden kann. Führen Sie den folgenden Befehl im aus AWS CLI , um sich beim AWS Access-Portal anzumelden.

```
aws sso login
```

Da Sie ein Standardprofil eingerichtet haben, müssen Sie den Befehl nicht mit einer `--profile` Option aufrufen. Wenn Ihre SSO-Token-Provider-Konfiguration ein benanntes Profil verwendet, lautet der Befehl`aws sso login --profile named-profile`.

Führen Sie den folgenden AWS CLI -Befehl aus, um zu testen, ob Sie bereits eine aktive Sitzung haben.

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

In der Antwort auf diesen Befehl sollten das in der freigegebenen `config`-Datei konfigurierte IAM-Identity-Center-Konto und der Berechtigungssatz angegeben werden.

> **Note**
>
> Wenn Sie bereits über eine aktive AWS -Zugriffsportalsitzung verfügen und `aws sso login` ausführen, müssen Sie keine Anmeldeinformationen angeben.
> Es wird jedoch ein Dialogfeld angezeigt, in dem Sie um Erlaubnis `botocore` zum Zugriff auf Ihre Informationen gebeten werden. `botocore`ist die Grundlage für die AWS CLI .
> Wählen Sie Zulassen aus, um den Zugriff auf Ihre Informationen für das AWS CLI und SDK für Kotlin zu autorisieren.

Installieren Sie Java und ein Build-Tool

Ihre Entwicklungsumgebung benötigt Folgendes:

- JDK 8 oder höher. Das AWS SDK für Kotlin funktioniert mit dem Oracle Java SE Development Kit und mit Distributionen von Open Java Development Kit (OpenJDK) wie Amazon CorrettoRed Hat OpenJDK und JDK. AdoptOpen
- Ein Build-Tool oder eine IDE, die Maven Central wie Apache Maven, Gradle oder IntelliJ unterstützt.
  - Informationen zur Installation und Verwendung von Maven finden Sie unter http://maven.apache.org/.
  - Informationen zur Installation und Verwendung von Gradle finden Sie unter https://gradle.org/.
  - Informationen zur Installation und Verwendung von IntelliJ IDEA finden Sie unter. https://www.jetbrains.com/idea/

Verwenden temporärer Anmeldeinformationen

Als Alternative zur Konfiguration des IAM Identity Center Single Sign-On-Zugriffs für das SDK können Sie Ihre Entwicklungsumgebung mit temporären Anmeldeinformationen konfigurieren.

Richten Sie eine lokale Anmeldeinformationsdatei für temporäre Anmeldeinformationen ein

1. Erstellen Sie eine gemeinsame Anmeldeinformationsdatei
2. Fügen Sie in die Anmeldeinformationsdatei den folgenden Platzhaltertext ein, bis Sie funktionierende temporäre Anmeldeinformationen einfügen:

```
[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. Speichern Sie die Datei. Die Datei `~/.aws/credentials` sollte jetzt auf Ihrem lokalen Entwicklungssystem vorhanden sein. Diese Datei enthält das [Standard-] Profil, das das SDK für Kotlin verwendet, wenn kein bestimmtes benanntes Profil angegeben ist.
4. Melden Sie sich beim AWS Zugangsportal an

5. Folgen Sie diesen Anweisungen unter der Überschrift Manuelle Aktualisierung der Anmeldeinformationen, um die Anmeldeinformationen für die IAM-Rolle aus dem AWS Zugriffsportal zu kopieren.

   a. Wählen Sie für Schritt 4 der verlinkten Anleitung den IAM-Rollennamen aus, der den Zugriff für Ihre Entwicklungsanforderungen gewährt. Diese Rolle hat normalerweise einen Namen wie PowerUserAccessoder Developer.

   b. Wählen Sie für Schritt 7 die Option Manuelles Hinzufügen eines Profils zu Ihrer AWS Anmeldeinformationsdatei und kopieren Sie den Inhalt.

6. Fügen Sie die kopierten Anmeldeinformationen in Ihre lokale `credentials` Datei ein und entfernen Sie den generierten Profilnamen. Ihre Datei sollte wie folgt aussehen:

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

7. Speichern Sie die `credentials` Datei

Das SDK für Kotlin greift auf diese temporären Anmeldeinformationen zu, wenn es einen Service-Client erstellt, und verwendet sie für jede Anfrage. Die in Schritt 5a ausgewählten Einstellungen für die IAM-Rolle bestimmen, wie lange die temporären Anmeldeinformationen gültig sind. Die maximale Dauer beträgt zwölf Stunden.

Wenn die temporären Anmeldeinformationen abgelaufen sind, wiederholen Sie die Schritte 4 bis 7.

Erstellen Sie Projekt-Build-Dateien

Nachdem Sie den Single Sign-On-Zugriff und Ihre Entwicklungsumgebung konfiguriert haben, erstellen Sie ein Kotlin-Projekt mit Ihrem bevorzugten Build-Tool. Geben Sie in der Build-Datei die Abhängigkeiten für die an AWS-Services , auf die Ihre Anwendung zugreifen muss.

Eine Liste aller möglichen Namen von Maven-Artefakten finden Sie in der API-Referenzdokumentation. Die neueste Version des SDK finden Sie in der neuesten Version unter. GitHub

Die folgenden Beispiel-Build-Dateien enthalten die notwendigen Elemente, um mit der Codierung eines Projekts mit sieben zu beginnen AWS-Services.

Gradle

The AWS SDK für Kotlin veröffentlicht einen Gradle-Versionskatalog und eine Stückliste (BOM), mit deren Hilfe Sie die Namen von Abhängigkeiten ermitteln und Versionsnummern für mehrere Artefakte synchronisieren können.

Beachten Sie, dass Versionskataloge eine Vorschaufunktion von Gradle vor Version 8 sind. Abhängig von der Version von Gradle, die Sie verwenden, müssen Sie sich möglicherweise über die Feature Preview API anmelden.

Um einen Gradle-Versionskatalog zu verwenden

1. Fügen Sie in Ihrer `settings.gradle.kts` Datei einen `versionCatalogs` Block innerhalb des `dependencyResolutionManagement` Blocks hinzu.

   Die folgende Beispieldatei konfiguriert den AWS SDK für Kotlin Versionskatalog. Sie können zu dem *`X.Y.Z`* Link navigieren, um die neueste verfügbare Version zu sehen.

   ```
   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. Deklarieren Sie Abhängigkeiten `build.gradle.kts` mithilfe der typsicheren Bezeichner, die im Versionskatalog verfügbar sind.

   Die folgende Beispieldatei deklariert Abhängigkeiten für sieben. AWS-Services

   ```
   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"
}
```

* Java-Version, zum Beispiel 17 oder21.

Maven

Die folgende `pom.xml` Beispieldatei hat Abhängigkeiten für sieben AWS-Services. Sie können zu dem *X.Y.Z* Link navigieren, um die neueste verfügbare Version zu sehen.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>

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

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

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

        </dependencies>
    </dependencyManagement>
```

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

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

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

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

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

* Java-Version, zum Beispiel 17 oder21.

Codieren Sie Ihr Kotlin-Projekt mit dem SDK für Kotlin

Jetzt beginnt der Spaß. Bei der Entwicklung Ihrer Anwendung finden Sie in der AWS SDK für Kotlin API-Referenz vollständige Informationen zu den API-Vorgängen. Verwenden Sie die folgenden Links für allgemeine Kotlin-API-Informationen:

- API-Referenz für die Standardbibliothek
- Überblick über Coroutines
- Coroutinen-API

Loggen Sie sich mit dem ein AWS CLI

Immer wenn Sie ein Programm ausführen, das zugreift AWS-Services, benötigen Sie eine aktive AWS Access-Portal-Sitzung. Sie erreichen dies mit dem folgenden -Befehl.

```
aws sso login
```

Da Sie ein Standardprofil eingerichtet haben, müssen Sie den Befehl nicht mit einer `--profile`-Option aufrufen. Wenn Ihre IAM Identity Center Single Sign-On-Konfiguration ein benanntes Profil verwendet, lautet der Befehl. `aws sso login --profile named-profile`

Führen Sie den folgenden AWS CLI Befehl aus, um zu testen, ob Sie bereits eine aktive Sitzung haben.

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

In der Antwort auf diesen Befehl sollten das in der freigegebenen `config`-Datei konfigurierte IAM-Identity-Center-Konto und der Berechtigungssatz angegeben werden.

Konfiguriere das AWS SDK für Kotlin

In diesem Abschnitt wird erklärt, wie Sie einen Service-Client mit dem konfigurieren AWS SDK für Kotlin. Weitere Informationen finden Sie im SDK- und Tools-Referenzhandbuch, das einen Überblick über die Konfiguration enthält, die für alle gilt AWS SDKs.

Inhalt

- Erstellen Sie einen Service-Client
  - Konfigurieren Sie einen Client im Code
  - Konfigurieren Sie einen Client aus der Umgebung
  - Schließen Sie den Client
- AWS-Region Auswahl
  - Anbieterkette in der Standardregion
- Anmeldeinformationsanbieter
  - Die standardmäßige Anbieterkette für Anmeldeinformationen
    - Erfahren Sie mehr über die Anbieterkette für Standardanmeldedaten
  - Geben Sie einen Anbieter für Anmeldeinformationen an
    - Anmeldeinformationen bei einem eigenständigen Anbieter zwischenspeichern
- Client-Endpunkte konfigurieren
  - Benutzerdefinierte Konfiguration
    - Legen Sie endpointUrl fest.
    - Legen Sie endpointProvider fest.
      - EndpointProvider-Eigenschaften
    - endpointUrl oder endpointProvider
    - Ein Hinweis zu Amazon S3
  - Beispiele
    - endpointUrlBeispiel für
    - endpointProviderBeispiel für
    - endpointUrl und endpointProvider
- HTTP
  - HTTP-Client-Konfiguration

- Basiskonfiguration
  - Importe
  - Code
- Erweiterte Konfiguration
  - Geben Sie einen HTTP-Engine-Typ an
    - Importe
    - Code
    - Verwenden der OkHttp4Engine
    - Verwenden Sie einen expliziten HTTP-Client
      - Importe
      - Code
  - Überwachung inaktiver Verbindungen
    - Importe
    - Code
- Verwenden eines HTTP-Proxys
  - Verwenden Sie die JVM-Systemeigenschaften
  - Verwenden Sie Umgebungsvariablen
  - Verwenden Sie einen Proxy auf EC2-Instances
- HTTP-Interzeptoren
  - Registrierung des Interceptors
    - Interceptor für alle Service-Client-Operationen
    - Interceptor nur für bestimmte Operationen
- Erzwingen Sie eine TLS-Mindestversion
  - Konfigurieren Sie die HTTP-Engine
  - Legen Sie die sdk.minTls JVM-Systemeigenschaft fest
  - Legen Sie die Umgebungsvariable fest SDK_MIN_TLS
- Wiederholungen im AWS SDK für Kotlin
  - Grundlegendes zum Verhalten bei Wiederholungen
    - Standardkonfiguration für Wiederholungsversuche
    - Welche Ausnahmen können wiederholt werden?

- Wiederholbar anhand des Fehlercodes
- Kann mit dem HTTP-Statuscode erneut versucht werden
- Kann je nach Fehlertyp erneut versucht werden
- Kann anhand von SDK-Metadaten erneut versucht werden
- Prüfen Sie, ob eine Ausnahme erneut versucht werden kann
- Welche Ausnahmen erreichen Ihren Code, wenn Wiederholungsversuche fehlschlagen
- Anpassen des Wiederholungsverhaltens
- Konfigurieren Sie die maximale Anzahl an Versuchen
- Konfigurieren Sie Verzögerungen und Backoff
- Konfigurieren Sie den Token-Bucket für Wiederholungsversuche
- Konfigurieren Sie adaptive Wiederholungsversuche
- Beobachtbarkeit
- Konfigurieren Sie ein TelemetryProvider
- Konfigurieren Sie den globalen Standardtelemetrie-Anbieter
- Konfigurieren Sie einen Telemetrieanbieter für einen bestimmten Dienstclient
- Kennzahlen
- Protokollierung
- Geben Sie den Protokollmodus für Nachrichten auf Kabelebene an
- Stellen Sie den Protokollmodus im Code ein
- Legt den Protokollmodus in der Umgebung fest
- Anbieter von Telemetriedaten
- Konfigurieren Sie den OpenTelemetry basierten Telemetrieanbieter
- Voraussetzungen
- Das SDKs konfigurieren
- Ressourcen
- Überschreiben Sie die Service-Client-Konfiguration
- Lebenszyklus eines überschriebenen Clients
- Ressourcen, die von Clients gemeinsam genutzt werden

Erstellen Sie einen Service-Client

Um eine Anfrage an zu stellen AWS-Service, müssen Sie zuerst einen Client für diesen Dienst instanziieren.

Sie können allgemeine Einstellungen für Dienstclients konfigurieren, z. B. den zu verwendenden HTTP-Client, die Protokollierungsebene und die Konfiguration erneut versuchen. Darüber hinaus benötigt jeder Service-Client einen AWS-Region und einen Anbieter für Anmeldeinformationen. Das SDK verwendet diese Werte, um Anfragen an die richtige Region zu senden und Anfragen mit den richtigen Anmeldeinformationen zu signieren.

Sie können diese Werte programmgesteuert im Code angeben oder sie automatisch aus der Umgebung laden lassen.

Konfigurieren Sie einen Client im Code

Um einen Service-Client mit bestimmten Werten zu konfigurieren, können Sie diese in einer Lambda-Funktion angeben, die an die Factory-Methode des Service Clients übergeben wird, wie im folgenden Codeausschnitt gezeigt.

```
val dynamoDbClient = DynamoDbClient {
    region = "us-east-1"
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
}
```

Alle Werte, die Sie nicht im Konfigurationsblock angeben, werden auf Standardwerte gesetzt. Wenn Sie beispielsweise keinen Anbieter für Anmeldeinformationen angeben, wie dies im vorherigen Code der Fall ist, verwendet der Anbieter für Anmeldeinformationen standardmäßig die Standardanbieterkette für Anmeldeinformationen.

> **Warning**
>
> Für einige Eigenschaften wie z. B. gibt `region` es keine Standardeinstellung. Sie müssen sie explizit im Konfigurationsblock angeben, wenn Sie die programmgesteuerte Konfiguration verwenden. Wenn das SDK die Eigenschaft nicht auflösen kann, schlagen API-Anfragen möglicherweise fehl.

Konfigurieren Sie einen Client aus der Umgebung

Beim Erstellen eines Service-Clients kann das SDK Standorte in der aktuellen Ausführungsumgebung überprüfen, um einige Konfigurationseigenschaften zu ermitteln. Zu diesen Speicherorten gehören gemeinsam genutzte Konfigurations- und Anmeldeinformationsdateien, Umgebungsvariablen und JVM-Systemeigenschaften. Zu den verfügbaren Eigenschaften, die behoben werden können, gehören AWS Region, Wiederholungsstrategie, Protokollmodus und andere. Weitere Informationen zu allen Einstellungen, die das SDK in der Ausführungsumgebung auflösen kann, finden Sie im Referenzhandbuch AWS SDKs und im Referenzhandbuch für Tools-Einstellungen.

Verwenden Sie die statische Methode `suspend fun fromEnvironment()` auf der Service-Client-Schnittstelle, um einen Client mit einer Konfiguration aus der Umgebung zu erstellen:

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

Das Erstellen eines Clients auf diese Weise ist nützlich, wenn er auf Amazon EC2 oder einem anderen Kontext ausgeführt wird AWS Lambda, in dem die Konfiguration eines Service-Clients in der Umgebung verfügbar ist. Dadurch wird Ihr Code von der Umgebung entkoppelt, in der er ausgeführt wird, und es wird einfacher, Ihre Anwendung in mehreren Regionen bereitzustellen, ohne den Code zu ändern.

Darüber hinaus können Sie bestimmte Eigenschaften überschreiben, indem Sie einen Lambda-Block an übergeben. `fromEnvironment` Das folgende Beispiel lädt einige Konfigurationseigenschaften aus der Umgebung (z. B. Region), überschreibt jedoch ausdrücklich den Anbieter für Anmeldeinformationen, um Anmeldeinformationen aus einem Profil zu verwenden.

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

Das SDK verwendet Standardwerte für jede Konfigurationseigenschaft, die nicht anhand der programmatischen Einstellungen oder der Umgebung bestimmt werden kann. Wenn Sie beispielsweise keinen Anbieter für Anmeldeinformationen im Code oder in einer Umgebungseinstellung angeben, verwendet der Anbieter für Anmeldeinformationen standardmäßig die Standardanbieterkette für Anmeldeinformationen.

> **Warning**
>
> Für einige Eigenschaften wie Region gibt es keine Standardeinstellung. Sie müssen sie in einer Umgebungseinstellung oder explizit im Konfigurationsblock angeben. Wenn das SDK die Eigenschaft nicht auflösen kann, schlagen API-Anfragen möglicherweise fehl.

> **Note**
>
> Eigenschaften im Zusammenhang mit Anmeldeinformationen — wie temporäre Zugriffsschlüssel und SSO-Konfiguration — sind zwar in der Ausführungsumgebung zu finden, die Werte stammen jedoch nicht vom Client zum Zeitpunkt der Erstellung. Stattdessen wird bei jeder Anfrage von der Credentials Provider-Ebene auf die Werte zugegriffen.

Schließen Sie den Client

Wenn Sie den Service-Client nicht mehr benötigen, schließen Sie ihn, um alle Ressourcen freizugeben, die er verwendet:

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

Da Service-Clients die `Closeable`Schnittstelle erweitern, können Sie die `use`Erweiterung verwenden, um den Client am Ende eines Blocks automatisch zu schließen, wie im folgenden Codeausschnitt gezeigt.

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

Im vorherigen Beispiel erhält der Lambda-Block einen Verweis auf den Client, der gerade erstellt wurde. Sie können Operationen mit dieser Client-Referenz aufrufen, und wenn der Block abgeschlossen ist — auch durch Auslösen einer Ausnahme — wird der Client geschlossen.

AWS-Region Auswahl

Mit AWS-Regionen können Sie auf Unternehmen zugreifen AWS-Services , die in einem bestimmten geografischen Gebiet tätig sind. Dies ist nicht nur für die Redundanz nützlich, sondern sorgt auch dafür, dass Ihre Daten und Anwendungen in der Nähe Ihres Standorts sowie des Standorts Ihrer Benutzer ausgeführt werden.

Anbieterkette in der Standardregion

Beim Laden der Konfiguration eines Service-Clients aus der Umgebung wird der folgende Suchvorgang verwendet:

1. Jede explizite Region, die im Builder festgelegt wurde.
2. Die `aws.region` JVM-Systemeigenschaft ist überprüft. Wenn sie gesetzt ist, wird diese Region in der Konfiguration des Clients verwendet.
3. Die Umgebungsvariable `AWS_REGION` wird geprüft. Wenn sie gesetzt ist, wird diese Region in der Konfiguration des Clients verwendet.
   a. Hinweis: Diese Umgebungsvariable wird vom Lambda-Container festgelegt.
4. Das SDK überprüft die AWS gemeinsam genutzte Konfigurationsdatei. Wenn die `region` Eigenschaft für das aktive Profil festgelegt ist, verwendet das SDK sie.
   a. Die Umgebungsvariable `AWS_CONFIG_FILE` kann verwendet werden, um den Speicherort der gemeinsam genutzten Konfigurationsdatei anzupassen.
   b. Die `aws.profile` JVM-Systemeigenschaft oder die `AWS_PROFILE` Umgebungsvariable können verwendet werden, um das Profil anzupassen, das das SDK lädt.
5. Das SDK versucht, den Amazon EC2 Instance Metadata Service zu verwenden, um die Region der aktuell laufenden EC2-Instance zu ermitteln.
6. Wenn die Region zu diesem Zeitpunkt immer noch nicht gelöst ist, schlägt die Client-Erstellung mit einer Ausnahme fehl.

Anmeldeinformationsanbieter

⚠ Die Reihenfolge, in der die standardmäßige Anbieterkette für Anmeldeinformationen Anmeldeinformationen auflöst, hat sich mit Version 1.4.0 geändert. Weitere Informationen finden Sie im nachfolgenden Hinweis.

Wenn Sie Anfragen mit dem an Amazon Web Services senden AWS SDK für Kotlin, müssen Anfragen kryptografisch signiert werden, wobei die Anmeldeinformationen von ausgestellt wurden. AWS Das Kotlin SDK signiert die Anfrage automatisch für Sie. Um die Anmeldeinformationen abzurufen, kann das SDK Konfigurationseinstellungen verwenden, die sich an verschiedenen Stellen befinden, z. B. JVM-Systemeigenschaften, Umgebungsvariablen, gemeinsam genutzte `credentials` Dateien AWS `config` und Dateien sowie Amazon EC2 EC2-Instance-Metadaten.

Das SDK verwendet die Abstraktion des Anbieters für Anmeldeinformationen, um das Abrufen von Anmeldeinformationen aus verschiedenen Quellen zu vereinfachen. Das SDK enthält mehrere Implementierungen von Credentials Provider.

Wenn die abgerufene Konfiguration beispielsweise IAM Identity Center-Einstellungen für den Single Sign-On-Zugriff aus der gemeinsam genutzten `config` Datei enthält, ruft das SDK zusammen mit dem IAM Identity Center temporäre Anmeldeinformationen ab, an die es Anfragen sendet. AWS-Services Bei diesem Ansatz zur Erfassung von Anmeldeinformationen verwendet das SDK den IAM Identity Center-Anbieter (auch bekannt als SSO-Anmeldeinformationsanbieter). Im Abschnitt zur Einrichtung dieses Handbuchs wurde diese Konfiguration beschrieben.

Um einen bestimmten Anbieter für Anmeldeinformationen zu verwenden, können Sie einen angeben, wenn Sie einen Service-Client erstellen. Alternativ können Sie die standardmäßige Anbieterkette für Anmeldeinformationen verwenden, um automatisch nach Konfigurationseinstellungen zu suchen.

Die standardmäßige Anbieterkette für Anmeldeinformationen

Wenn es bei der Client-Erstellung nicht explizit angegeben wird, verwendet das SDK für Kotlin einen Anbieter für Anmeldeinformationen, der nacheinander jeden Ort überprüft, an dem Sie Anmeldeinformationen angeben können. Dieser Standardanmeldeanmeldeanbieter ist als eine Kette von Anbietern von Anmeldeinformationen implementiert.

Um die Standardkette für die Bereitstellung von Anmeldeinformationen in Ihrer Anwendung zu verwenden, erstellen Sie einen Dienstclient, ohne explizit eine `credentialsProvider` Eigenschaft anzugeben.

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

Weitere Informationen zur Erstellung von Dienstclients finden Sie unter Konstruieren und Konfigurieren eines Clients.

Erfahren Sie mehr über die Anbieterkette für Standardanmeldedaten

Die standardmäßige Anbieterkette für Anmeldeinformationen sucht anhand der folgenden vordefinierten Reihenfolge nach der Konfiguration von Anmeldeinformationen. Wenn die konfigurierten Einstellungen gültige Anmeldeinformationen bereitstellen, wird die Kette beendet.

1. AWS Zugriffstasten (JVM-Systemeigenschaften)

   Das SDK sucht nach den `aws.sessionToken` JVM-Systemeigenschaften `aws.accessKeyIdaws.secretAccessKey`, und.

2. AWS Zugriffstasten (Umgebungsvariablen)

   Das SDK versucht, Anmeldeinformationen aus den `AWS_ACCESS_KEY_ID` `AWS_SESSION_TOKEN` Umgebungsvariablen und zu laden. `AWS_SECRET_ACCESS_KEY`

3. Web-Identitätstoken

   Das SDK sucht nach den Umgebungsvariablen `AWS_WEB_IDENTITY_TOKEN_FILE` und/oder `AWS_ROLE_ARN` (oder den JVM-Systemeigenschaften `aws.webIdentityTokenFile` und`aws.roleArn`). Basierend auf den Token-Informationen und der Rolle ruft das SDK temporäre Anmeldeinformationen ab.

4. Ein Profil in einer Konfigurationsdatei

   In diesem Schritt verwendet das SDK Einstellungen, die einem Profil zugeordnet sind. Standardmäßig verwendet das SDK gemeinsam genutzte `credentials` Dateien AWS `config` und Dateien. Wenn die `AWS_CONFIG_FILE` Umgebungsvariable jedoch festgelegt ist, verwendet das SDK diesen Wert. Wenn die `AWS_PROFILE` Umgebungsvariable (oder die `aws.profile` JVM-Systemeigenschaft) nicht gesetzt ist, sucht das SDK nach dem „Standard“ -Profil, andernfalls sucht es nach dem Profil, das dem `AWS_PROFILE`'s Wert entspricht.

   Das SDK sucht auf der Grundlage der im vorherigen Absatz beschriebenen Konfiguration nach dem Profil und verwendet die dort definierten Einstellungen. Wenn die vom SDK gefundenen Einstellungen eine Mischung aus Einstellungen für verschiedene Ansätze von Anmeldeinformationsanbietern enthalten, verwendet das SDK die folgende Reihenfolge:

   a. AWS Zugriffstasten (Konfigurationsdatei) — Das SDK verwendet die Einstellungen für `aws_access_key_idaws_access_key_id`, und. `aws_session_token`

   b. Rollenkonfiguration annehmen — Wenn das SDK `credential_source` Einstellungen `source_profile` und/oder findet`role_arn`, versucht es, eine Rolle anzunehmen. Wenn das SDK die `source_profile` Einstellung findet, bezieht es Anmeldeinformationen aus

einem anderen Profil, um temporäre Anmeldeinformationen für die von angegebene Rolle zu erhaltenrole_arn. Wenn das SDK die credential_source Einstellung findet, bezieht es die Anmeldeinformationen je nach Wert der credential_source Einstellung aus einem Amazon ECS-Container, einer Amazon EC2-Instance oder aus Umgebungsvariablen. Anschließend verwendet es diese Anmeldeinformationen, um temporäre Anmeldeinformationen für die Rolle abzurufen.

Ein Profil sollte entweder die source_profile Einstellung oder die credential_source Einstellung enthalten, aber nicht beides.

c. Konfiguration des Web-Identitätstokens — Wenn das SDK web_identity_token_file Einstellungen findetrole_arn, ruft es temporäre Anmeldeinformationen für den Zugriff auf AWS Ressourcen ab, die auf dem role_arn und dem Token basieren.

d. SSO-Token-Konfiguration — Wenn das SDK sso_role_name Einstellungen (zusammen mit einem zugehörigen sso-session Abschnitt in den Konfigurationsdateien) findetsso_session, ruft das SDK temporäre Anmeldeinformationen vom IAM Identity Center-Dienst ab. sso_account_id

e. Legacy-SSO-Konfiguration — Wenn das SDK sso_role_name Einstellungen sso_start_url,sso_region, und findetsso_account_id, ruft das SDK temporäre Anmeldeinformationen vom IAM Identity Center-Dienst ab.

f. Anmeldekonfiguration — Wenn das SDK eine login_session Einstellung findet, verwendet es die temporären Anmeldeinformationen aus der Anmeldesitzung oder versucht, sie zu aktualisieren, wenn sie in weniger als 5 Minuten ablaufen. Informationen zum Starten einer Anmeldesitzung finden Sie im AWS CLI-Benutzerhandbuch.

g. Prozesskonfiguration — Wenn das SDK eine credential_process Einstellung findet, verwendet es den Pfadwert, um einen Prozess aufzurufen und temporäre Anmeldeinformationen abzurufen.

5. Anmeldeinformationen für den Container

Das SDK sucht nach Umgebungsvariablen AWS_CONTAINER_CREDENTIALS_RELATIVE_URI oder AWS_CONTAINER_CREDENTIALS_FULL_URI und AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE oderAWS_CONTAINER_AUTHORIZATION_TOKEN. Es verwendet diese Werte, um Anmeldeinformationen über eine GET-Anfrage vom angegebenen HTTP-Endpunkt zu laden.

6. IMDS-Anmeldeinformationen

   Das SDK versucht, Anmeldeinformationen vom Instanz-Metadatendienst auf dem standardmäßigen oder konfigurierten HTTP-Endpunkt abzurufen. Das SDK unterstützt IMDSv2nur.

Wenn die Anmeldeinformationen zu diesem Zeitpunkt immer noch nicht geklärt sind, schlägt die Client-Erstellung mit einer Ausnahme fehl.

> **Hinweis: Ändern Sie die Reihenfolge der Auflösung der Anmeldeinformationen**
>
> Die oben beschriebene Reihenfolge der Auflösung von Anmeldeinformationen ist für die `1.4.x+` Veröffentlichung des SDK für Kotlin aktuell. Vor der `1.4.0` Veröffentlichung wurden die Elemente Nummer 3 und 4 vertauscht, und das aktuelle 4a-Element folgte auf das aktuelle 4g-Element.

Geben Sie einen Anbieter für Anmeldeinformationen an

Sie können einen Anbieter für Anmeldeinformationen angeben, anstatt die Standardanbieterkette zu verwenden. Dieser Ansatz gibt Ihnen direkte Kontrolle darüber, welche Anmeldeinformationen das SDK verwendet.

Um beispielsweise die Anmeldeinformationen für eine angenommene IAM-Rolle zu verwenden, geben Sie `StsAssumeRoleCredentialsProvider` bei der Erstellung des Clients Folgendes an:

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

Sie können auch eine benutzerdefinierte Kette (`CredentialsProviderChain`) erstellen, die mehrere Anbieter in Ihrer bevorzugten Reihenfolge kombiniert.

Anmeldeinformationen bei einem eigenständigen Anbieter zwischenspeichern

> **Important**
>
> In der Standardkette werden Anmeldeinformationen automatisch zwischengespeichert. Eigenständige Anbieter speichern Anmeldeinformationen nicht im Cache. Um zu vermeiden, dass bei jedem API-Aufruf Anmeldeinformationen abgerufen werden, umschließen Sie Ihren Anbieter mit einem`CachedCredentialsProvider`. Der zwischengespeicherte Anbieter ruft neue Anmeldeinformationen nur ab, wenn die aktuellen ablaufen.

Verwenden Sie die Klasse, um Anmeldeinformationen bei einem eigenständigen Anbieter zwischenzuspeichern`CachedCredentialsProvider`:

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

Verwenden Sie alternativ die `cached()` Erweiterungsfunktion für einen präziseren Code:

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

Client-Endpunkte konfigurieren

Wenn der AWS SDK für Kotlin aufruft AWS-Service, besteht einer der ersten Schritte darin, zu bestimmen, wohin die Anfrage weitergeleitet werden soll. Dieser Vorgang wird als Endpunktauflösung bezeichnet.

Sie können die Endpunktauflösung für das SDK konfigurieren, wenn Sie einen Service-Client erstellen. Die Standardkonfiguration für die Endpunktauflösung ist normalerweise in Ordnung, aber es gibt mehrere Gründe, die Sie dazu veranlassen könnten, die Standardkonfiguration zu ändern. Zwei Beispielgründe lauten wie folgt:

- Stellen Sie Anfragen an eine Vorabversion eines Dienstes oder an eine lokale Bereitstellung eines Dienstes.
- Zugriff auf bestimmte Servicefunktionen, die noch nicht im SDK modelliert wurden.

> **Warning**
>
> Die Endpunktauflösung ist ein SDK-Thema für Fortgeschrittene. Wenn Sie die Standardeinstellungen ändern, riskieren Sie, dass Ihr Code beschädigt wird. Die Standardeinstellungen sollten für die meisten Benutzer in Produktionsumgebungen gelten.

Benutzerdefinierte Konfiguration

Sie können die Endpunktauflösung eines Service-Clients mit zwei Eigenschaften anpassen, die bei der Erstellung des Clients verfügbar sind:

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

Legen Sie **`endpointUrl`** fest.

Sie können einen Wert für festlegen`endpointUrl`, um einen „Basis-Hostnamen“ für den Dienst anzugeben. Dieser Wert ist jedoch nicht endgültig, da er als Parameter an die `EndpointProvider` Instanz des Clients übergeben wird. Die `EndpointProvider` Implementierung kann diesen Wert dann überprüfen und möglicherweise ändern, um den endgültigen Endpunkt zu bestimmen.

Wenn Sie beispielsweise einen `endpointUrl` Wert für einen Amazon Simple Storage Service (Amazon S3) -Client angeben und einen `GetObject` Vorgang ausführen, fügt die standardmäßige Implementierung des Endpunktanbieters den Bucket-Namen in den Hostname-Wert ein.

In der Praxis legen Benutzer einen `endpointUrl` Wert fest, der auf eine Entwicklungs- oder Vorschauinstanz eines Dienstes verweist.

Legen Sie **`endpointProvider`** fest.

Die `EndpointProvider` Implementierung eines Serviceclients bestimmt die endgültige Endpunktauflösung. Die im folgenden Codeblock gezeigte `EndpointProvider` Schnittstelle macht die `resolveEndpoint` Methode verfügbar.

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

Ein Service-Client ruft die `resolveEndpoint` Methode für jede Anfrage auf. Der Service-Client verwendet den vom Anbieter zurückgegebenen `Endpoint` Wert ohne weitere Änderungen.

`EndpointProvider`-Eigenschaften

Die `resolveEndpoint` Methode akzeptiert ein dienstspezifisches `EndpointParameters` Objekt, das Eigenschaften enthält, die bei der Endpunktauflösung verwendet werden.

Jeder Dienst umfasst die folgenden Basiseigenschaften.

Name	Typ	Description
`region`	Zeichenfolge	Die AWS Region des Kunden
`endpoint`	Zeichenfolge	Eine Zeichenkettendarstellung des Wertesatzes von `endpointUrl`
`useFips`	Boolesch	Ob FIPS-Endpunkte in der Konfiguration des Clients aktiviert sind
`useDualStack`	Boolesch	Ob Dual-Stack-Endpunkte in der Konfiguration des Clients aktiviert sind

Dienste können zusätzliche Eigenschaften angeben, die für die Lösung erforderlich sind. Amazon S3 `S3EndpointParameters`enthält beispielsweise den Bucket-Namen und auch mehrere Amazon S3-spezifische Funktionseinstellungen. Die `forcePathStyle` Eigenschaft bestimmt beispielsweise, ob die virtuelle Host-Adressierung verwendet werden kann.

Wenn Sie Ihren eigenen Anbieter implementieren, sollten Sie keine eigene Instanz von erstellen müssen`EndpointParameters`. Das SDK stellt die Eigenschaften für jede Anfrage bereit und übergibt sie an Ihre Implementierung von`resolveEndpoint`.

endpointUrl oder **endpointProvider**

Es ist wichtig zu verstehen, dass die folgenden beiden Aussagen NICHT zu Clients mit gleichwertigem Verhalten bei der Endpunktauflösung führen:

```
// 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")
    }
}
```

Die Anweisung, mit der die `endpointUrl` Eigenschaft festgelegt wird, gibt eine Basis-URL an, die an den (Standard-) Anbieter übergeben wird und im Rahmen der Endpunktauflösung geändert werden kann.

Die Anweisung, die festlegt, `endpointProvider` gibt die endgültige URL an, die `S3Client` verwendet wird.

Sie können zwar beide Eigenschaften festlegen, in den meisten Fällen, in denen eine Anpassung erforderlich ist, geben Sie jedoch eine davon an. Als allgemeiner SDK-Benutzer geben Sie in den meisten Fällen einen `endpointUrl` Wert an.

Ein Hinweis zu Amazon S3

Amazon S3 ist ein komplexer Service, bei dem viele seiner Funktionen durch maßgeschneiderte Endpunktanpassungen, wie z. B. virtuelles Bucket-Hosting, modelliert wurden. Virtuelles Hosting ist eine Funktion von Amazon S3, bei der der Bucket-Name in den Hostnamen eingefügt wird.

Aus diesem Grund empfehlen wir, die `EndpointProvider` Implementierung in einem Amazon S3 S3-Serviceclient nicht zu ersetzen. Wenn Sie das Auflösungsverhalten erweitern müssen, indem Sie beispielsweise Anfragen an einen lokalen Entwicklungsstapel mit zusätzlichen Überlegungen zu Endpunkten senden, empfehlen wir, die Standardimplementierung zu verpacken. Das folgende `endpointProvider` Beispiel zeigt eine Beispielimplementierung dieses Ansatzes.

Beispiele

endpointUrlBeispiel für

Der folgende Codeausschnitt zeigt, wie der allgemeine Service-Endpunkt für einen Amazon S3 S3-Client außer Kraft gesetzt werden kann.

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

endpointProviderBeispiel für

Der folgende Codeausschnitt zeigt, wie Sie einen benutzerdefinierten Endpunktanbieter bereitstellen, der die Standardimplementierung für Amazon S3 umschließt.

```
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 und **endpointProvider**

Das folgende Beispielprogramm demonstriert die Interaktion zwischen den `endpointUrl` Einstellungen und. `endpointProvider` Dies ist ein Anwendungsfall für Fortgeschrittene.

```
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

Dieser Abschnitt behandelt die Konfiguration von HTTP-bezogenen Einstellungen in der. AWS SDK für Kotlin

Themen

- HTTP-Client-Konfiguration
- Verwenden eines HTTP-Proxys
- HTTP-Interzeptoren
- Erzwingen Sie eine TLS-Mindestversion

HTTP-Client-Konfiguration

Standardmäßig AWS SDK für Kotlin verwendet der einen HTTP-Client, der auf basiert OkHttp. Sie können den HTTP-Client und seine Konfiguration überschreiben, indem Sie einen explizit konfigurierten Client angeben.

> **Warning**
>
> Unabhängig davon, welche HTTP-Engine Sie verwenden, können andere Abhängigkeiten in Ihrem Projekt transitive Abhängigkeiten aufweisen, die mit der spezifischen Engine-Version in Konflikt stehen, die für das SDK erforderlich ist. Insbesondere Frameworks wie Spring Boot sind dafür bekannt, Abhängigkeiten wie das SDK zu verwalten OkHttp und sich auf ältere Versionen als das SDK zu verlassen. Weitere Informationen the section called “Wie löse ich Abhängigkeitskonflikte?” finden Sie unter.

> **Note**
>
> Standardmäßig verwendet jeder Service-Client seine eigene Kopie eines HTTP-Clients. Wenn Sie in Ihrer Anwendung mehrere Dienste verwenden, möchten Sie möglicherweise einen einzelnen HTTP-Client erstellen und ihn für alle Dienstclients gemeinsam nutzen.

Basiskonfiguration

Wenn Sie einen Service-Client konfigurieren, können Sie den Standard-Engine-Typ konfigurieren. Das SDK verwaltet die resultierende HTTP-Client-Engine und schließt sie automatisch, wenn sie nicht mehr benötigt wird.

Das folgende Beispiel zeigt die Konfiguration eines HTTP-Clients während der Initialisierung eines DynamoDB-Clients.

Importe

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

Code

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

```
}.use { ddb ->

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

Erweiterte Konfiguration

Die Standard-HTTP-Konfiguration ist für die meisten Anwendungsfälle geeignet. Für einige erweiterte Anwendungsfälle, wie z. B. Umgebungen mit hohem Durchsatz, bieten die folgenden erweiterten Konfigurationsoptionen zusätzliche Funktionen und Möglichkeiten:

Geben Sie einen HTTP-Engine-Typ an

Um einen nicht standardmäßigen HTTP-Engine-Typ anzugeben oder um die Konfiguration anzupassen, die für einen bestimmten HTTP-Engine-Typ spezifisch ist, können Sie einen zusätzlichen Parameter übergeben, der den Engine-Typ `httpClient` angibt.

Das folgende Beispiel gibt den an `OkHttpEngine`, den Sie zur Konfiguration der `maxConcurrencyPerHost`Eigenschaft verwenden können.

Importe

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

Code

```
DynamoDbClient {
    region = "us-east-2"
    httpClient(OkHttpEngine) {  // The first parameter specifies the HTTP engine type.
        // The following parameter is generic HTTP configuration available in any
 engine type.
        maxConcurrency = 64u

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

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

Die möglichen Werte für den Motortyp sind `OkHttpEngine` `OkHttp4Engine`, und `CrtHttpEngine`.

Um für eine HTTP-Engine spezifische Konfigurationsparameter zu verwenden, müssen Sie die Engine als Abhängigkeit zur Kompilierzeit hinzufügen. Für die `OkHttpEngine` fügen Sie mit Gradle die folgende Abhängigkeit hinzu.

(Sie können zum *`X.Y.Z`* Link navigieren, um die neueste verfügbare Version zu sehen.)

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

Fügen Sie für `CrtHttpEngine` die die folgende Abhängigkeit hinzu.

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

Verwenden der **`OkHttp4Engine`**

Verwenden Sie die`OkHttp4Engine`, wenn Sie die Standardeinstellung nicht verwenden können`OkHttpEngine`. Das GitHub Smithy-Kotlin-Repository enthält Informationen darüber, wie Sie das konfigurieren und verwenden. `OkHttp4Engine`

Verwenden Sie einen expliziten HTTP-Client

Wenn Sie einen expliziten HTTP-Client verwenden, sind Sie für dessen Lebensdauer verantwortlich, einschließlich dessen Schließung, wenn Sie ihn nicht mehr benötigen. Ein HTTP-Client muss mindestens so lange aktiv sein wie jeder Service-Client, der ihn verwendet.

Das folgende Codebeispiel zeigt Code, der dafür sorgt, dass der HTTP-Client aktiv bleibt, solange der aktiv `DynamoDbClient` ist. Die `use`Funktion stellt sicher, dass der HTTP-Client ordnungsgemäß geschlossen wird.

Importe

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

Code

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

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

Überwachung inaktiver Verbindungen

> **Important**
>
> Die Funktion zum Abrufen von Verbindungen im Leerlauf (`connectionIdlePollingInterval`) der OkHttp Engine wurde durch automatische Wiederholungsversuche bei Verbindungsfehlern () ersetzt. `retryOnConnectionFailure` Das Abrufen inaktiver Verbindungen ist in SDK-Version v1.7 veraltet und wird in SDK-Version v1.8 entfernt. Weitere Informationen finden Sie im entsprechenden GitHub Diskussionsbeitrag.

`OkHttpEngine`bietet die `connectionIdlePollingInterval`Konfigurationsoption zur Überwachung inaktiver Verbindungen im Hinblick auf das Schließen per Fernzugriff. Diese Funktion erkennt, wenn Dienste Verbindungen geschlossen haben, die sich noch im Verbindungspool befinden, und verhindert so Fehler bei nachfolgenden Anfragen.

Wenn der Wert auf einen Wert ungleich Null gesetzt `connectionIdlePollingInterval` ist, fragt die Engine Verbindungen, die freigegeben wurden, zurück zum Verbindungspool. Bei der Abfrage werden Lesevorgänge blockiert, wobei das Socket-Timeout auf das angegebene Intervall eingestellt ist. Die Abfrage wird automatisch abgebrochen, wenn die Engine die Verbindung aus dem Pool bezieht oder wenn die Verbindung gelöscht und geschlossen wird.

Wenn dieser Wert `null` (Standard) ist, ist die Abfrage deaktiviert. Bei inaktiven Verbindungen im Pool, die remote geschlossen wurden, können Fehler auftreten, wenn sie für nachfolgende Aufrufe abgerufen werden.

> **Note**
>
> Da die Abfrageschleife blockierende Lesevorgänge verwendet, können Engine-Aufrufe zum Herstellen oder Schließen einer Verbindung um das `connectionIdlePollingInterval` Intervall verzögert werden. Wenn Sie für das Intervall einen niedrigen Wert wählen, nimmt das SDK schneller Verbindungen auf, allerdings auf Kosten einer höheren Ressourcenauslastung im Leerlauf.

Importe

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.smithy.kotlin.runtime.http.engine.okhttp.OkHttpEngine
import kotlin.time.Duration.Companion.milliseconds
```

Code

```
S3Client.fromEnvironment {
    httpEngine(OkHttpEngine) {
        connectionIdlePollingInterval = 50.milliseconds
    }
}.use { s3 ->
  // Use the Amazon S3 client
}
```

Verwenden eines HTTP-Proxys

Für den Zugriff AWS über Proxyserver mit dem AWS SDK für Kotlin können Sie entweder JVM-Systemeigenschaften oder Umgebungsvariablen konfigurieren. Wenn beide angegeben werden, haben die JVM-Systemeigenschaften Vorrang.

Verwenden Sie die JVM-Systemeigenschaften

Das SDK sucht nach den JVM-Systemeigenschaften`https.proxyHost`,`https.proxyPort`, und. `http.nonProxyHosts` Weitere Informationen zu diesen allgemeinen JVM-Systemeigenschaften finden Sie in der Java-Dokumentation unter Netzwerke und Proxys.

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

Verwenden Sie Umgebungsvariablen

Das SDK sucht nach den `no_proxy` Umgebungsvariablen `https_proxy``http_proxy`, und (und jeweils in Großbuchstaben geschriebenen Versionen).

```
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
```

Verwenden Sie einen Proxy auf EC2-Instances

Wenn Sie einen Proxy auf einer EC2-Instance konfigurieren, die mit einer angehängten IAM-Rolle gestartet wurde, achten Sie darauf, die Adresse auszunehmen, die für den Zugriff auf die Instance-Metadaten verwendet wird. Setzen Sie dazu die `http.nonProxyHosts` JVM-Systemeigenschaft oder `no_proxy` Umgebungsvariable auf die IP-Adresse des Instanz-Metadatendienstes, d. h. `169.254.169.254` Diese Adresse variiert nicht.

```
export no_proxy=169.254.169.254
```

HTTP-Interzeptoren

Sie können Interzeptoren verwenden, um sich an der Ausführung von API-Anfragen und -Antworten zu beteiligen. Interzeptoren sind offene Mechanismen, bei denen das SDK Code aufruft, den Sie schreiben, um Verhalten in den Lebenszyklus einzufügen. request/response Auf diese Weise können Sie eine In-Flight-Anfrage ändern, die Anforderungsverarbeitung debuggen, Ausnahmen anzeigen und vieles mehr.

Das folgende Beispiel zeigt einen einfachen Interceptor, der allen ausgehenden Anfragen einen zusätzlichen Header hinzufügt, bevor die Wiederholungsschleife aufgerufen wird.

```
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()
    }
}
```

Weitere Informationen und die verfügbaren Interception-Hooks finden Sie in der Interceptor-Schnittstelle.

Registrierung des Interceptors

Sie registrieren Interzeptoren, wenn Sie einen Service-Client erstellen oder wenn Sie die Konfiguration für eine bestimmte Gruppe von Operationen überschreiben.

Interceptor für alle Service-Client-Operationen

Der folgende Code fügt der Interceptors-Eigenschaft des Builders eine `AddHeader` Instanz hinzu. Dieser Zusatz fügt allen Operationen den `x-foo-version` Header hinzu, bevor die Wiederholungsschleife aufgerufen wird.

```
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 nur für bestimmte Operationen

Mithilfe der `withConfig` Erweiterung können Sie die Service-Client-Konfiguration für einen oder mehrere Operationen für einen beliebigen Service-Client außer Kraft setzen. Mit dieser Funktion können Sie zusätzliche Interzeptoren für eine Teilmenge von Operationen registrieren.

Das folgende Beispiel überschreibt die Konfiguration der `s3` Instanz für Operationen innerhalb der Erweiterung. `use` Die aufgerufenen Operationen `s3Scoped` enthalten `x-foo-version` sowohl die als auch die `x-bar-version` Header.

```
// '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.
}
```

Erzwingen Sie eine TLS-Mindestversion

Mit dem können Sie die TLS-Mindestversion konfigurieren AWS SDK für Kotlin, wenn Sie eine Verbindung zu Dienstendpunkten herstellen. Das SDK bietet verschiedene Konfigurationsoptionen. In der Reihenfolge von höchster bis niedrigster Priorität stehen folgende Optionen zur Verfügung:

- Konfigurieren Sie die HTTP-Engine explizit
- Legen Sie die `sdk.minTls` JVM-Systemeigenschaft fest
- Legen Sie die `SDK_MIN_TLS` Umgebungsvariable fest

Konfigurieren Sie die HTTP-Engine

Wenn Sie eine nicht standardmäßige HTTP-Engine für einen Service-Client angeben, können Sie das `tlsContext.minVersion` Feld festlegen.

Im folgenden Beispiel werden die HTTP-Engine und alle Service-Clients, die sie verwenden, so konfiguriert, dass sie mindestens TLS v1.2 verwenden.

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

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

Legen Sie die **`sdk.minTls`** JVM-Systemeigenschaft fest

Sie können die `sdk.minTls` JVM-Systemeigenschaft festlegen. Wenn Sie eine Anwendung starten, bei der die Systemeigenschaft festgelegt ist, AWS SDK für Kotlin verwenden alle HTTP-Engines, die

mit dieser Methode erstellt wurden, standardmäßig die angegebene TLS-Mindestversion. Sie können dies jedoch in der HTTP-Engine-Konfiguration explizit überschreiben. Die zulässigen Werte sind:

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

Legen Sie die Umgebungsvariable fest **`SDK_MIN_TLS`**

Sie können die `SDK_MIN_TLS` Umgebungsvariable festlegen. Wenn Sie eine Anwendung starten, bei der die Umgebungsvariable gesetzt ist, AWS SDK für Kotlin verwenden alle HTTP-Engines, die mit der angegebenen TLS-Mindestversion erstellt wurden, sofern sie nicht durch eine andere Option überschrieben wird.

Die zulässigen Werte sind:

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

Wiederholungen im AWS SDK für Kotlin

Aufrufe geben AWS-Services gelegentlich unerwartete Ausnahmen zurück. Bestimmte Arten von Fehlern, wie Drosselungen oder vorübergehende Fehler, können erfolgreich sein, wenn der Aufruf erneut versucht wird.

Auf dieser Seite wird beschrieben, wie Wiederholungsversuche automatisch AWS SDK für Kotlin verarbeitet werden und wie Sie das Wiederholungsverhalten für Ihre Anwendungen anpassen können.

Grundlegendes zum Verhalten bei Wiederholungen

In den folgenden Abschnitten wird erklärt, wie das SDK festlegt, wann Anfragen wiederholt werden müssen und welche Ausnahmen als Wiederholungsversuche gelten.

Standardkonfiguration für Wiederholungsversuche

Standardmäßig wird jeder Service-Client automatisch mit einer standardmäßigen Wiederholungsstrategie konfiguriert. In der Standardkonfiguration wird ein Anruf versucht, der bis zu dreimal fehlschlägt (der erste Versuch plus zwei Wiederholungsversuche). Die dazwischenliegende Verzögerung zwischen den einzelnen Aufrufen ist mit exponentiellem Backoff und zufälligem Jitter konfiguriert, um Wiederholungsversuche zu vermeiden. Diese Konfiguration funktioniert für die meisten Anwendungsfälle, kann jedoch unter bestimmten Umständen ungeeignet sein, z. B. bei Systemen mit hohem Durchsatz.

Das SDK versucht es nur bei Fehlern, die wiederholt werden können. Beispiele für Fehler, die wiederholt werden können, sind Socket-Timeouts, dienstseitige Drosselung, gleichzeitige oder optimistische Sperren sowie vorübergehende Dienstfehler. Fehlende oder ungültige Parameter, authentication/security Fehler und Ausnahmen bei Fehlkonfigurationen gelten nicht als wiederholbar.

Sie können die standardmäßige Wiederholungsstrategie anpassen, indem Sie die maximale Anzahl an Versuchen, Verzögerungen und Backoffs sowie die Token-Bucket-Konfiguration festlegen.

Welche Ausnahmen können wiederholt werden?

Der AWS SDK für Kotlin verwendet eine vorkonfigurierte Wiederholungsrichtlinie, die festlegt, welche Ausnahmen wiederholt werden können. Die Service-Client-Konfiguration verfügt über eine `retryPolicy` Eigenschaft, die angibt, welche Richtlinie auf Wiederholungsversuche angewendet wird. Wenn kein benutzerdefinierter Wert angegeben ist, ist AwsRetryPolicyder Standardwert.

Die folgenden Ausnahmen werden als wiederholbar eingestuft durch: `AwsRetryPolicy`

Wiederholbar anhand des Fehlercodes

Alle `ServiceException` mit einem `sdkErrorMetadata.errorCode` von:

- `BandwidthLimitExceeded`
- `EC2ThrottledException`
- `IDPCommunicationError`
- `LimitExceededException`
- `PriorRequestNotComplete`
- `ProvisionedThroughputExceededException`
- `RequestLimitExceeded`

- `RequestThrottled`
- `RequestThrottledException`
- `RequestTimeout`
- `RequestTimeoutException`
- `SlowDown`
- `ThrottledException`
- `Throttling`
- `ThrottlingException`
- `TooManyRequestsException`
- `TransactionInProgressException`

Kann mit dem HTTP-Statuscode erneut versucht werden

Alle `ServiceException` mit einem `sdkErrorMetadata.statusCode` von:

- 500 (Interner Servicefehler)
- 502 (Schlechtes Gateway)
- 503 (Dienst nicht verfügbar)
- 504 (Gateway-Timeout)

Kann je nach Fehlertyp erneut versucht werden

Alle `ServiceException` mit einem `sdkErrorMetadata.errorType` von:

- `ErrorType.Server`(z. B. interne Servicefehler)
- `ErrorType.Client`(z. B. eine ungültige Anfrage, eine nicht gefundene Ressource, verweigerter Zugriff usw.)

Kann anhand von SDK-Metadaten erneut versucht werden

IrgendwoSdkBaseException:

- `sdkErrorMetadata.isRetryable`ist `true` (z. B. ein clientseitiges Timeout, ein networking/socket Fehler usw.)
- `sdkErrorMetadata.isThrottling`ist `true` (z. B. zu viele Anfragen in kurzer Zeit stellen)

Eine vollständige Liste der Ausnahmen, die von jedem Service-Client ausgelöst werden können, finden Sie in der dienstspezifischen API-Referenzdokumentation.

Prüfen Sie, ob eine Ausnahme erneut versucht werden kann

Um festzustellen, ob das SDK eine Ausnahme als wiederholbar erachtet, überprüfen Sie die Eigenschaft für abgefangene Ausnahmen: `isRetryable`

```
try {
    dynamoDbClient.putItem {
        tableName = "MyTable"
        item = mapOf("id" to AttributeValue.S("123"))
    }
} catch (e: SdkBaseException) {
    println("Exception occurred: ${e.message}")

    if (e.sdkErrorMetadata.isRetryable) {
        println("This exception is retryable - SDK will automatically retry")
        println("If you're seeing this, retries may have been exhausted")
    } else {
        println("This exception is not retryable - fix the underlying issue")

        // Common non-retryable scenarios.
        when {
            e.message?.contains("ValidationException") == true ->
                println("Check your request parameters")
            e.message?.contains("AccessDenied") == true ->
                println("Check your IAM permissions")
            e.message?.contains("ResourceNotFound") == true ->
                println("Verify the resource exists")
        }
    }
}
```

Welche Ausnahmen erreichen Ihren Code, wenn Wiederholungsversuche fehlschlagen

Wenn der Wiederholungsmechanismus des SDK ein Problem nicht lösen kann, werden Ausnahmen in Ihrem Anwendungscode ausgelöst. Wenn Sie diese Ausnahmetypen verstehen, können Sie eine angemessene Fehlerbehandlung implementieren. Dies sind nicht die Ausnahmen, die Wiederholungen auslösen — diese werden intern vom SDK behandelt.

Ihr Code erfasst die catch Arten von Ausnahmen, wenn die Wiederholungsversuche erschöpft oder deaktiviert sind:

Serviceausnahmen nach Erschöpfung der Wiederholungsversuche

Wenn alle Wiederholungsversuche fehlschlagen, fängt Ihr Code die letzte Serviceausnahme (Unterklasse von`AwsServiceException`) ab, die zum Fehlschlagen des letzten Wiederholungsversuchs geführt hat. Dies kann ein Drosselungsfehler, ein Serverfehler oder eine andere dienstspezifische Ausnahme sein, die das SDK nicht durch Wiederholungen lösen konnte.

Netzwerkausnahmen nach Erschöpfung des Wiederholungsversuchs

Wenn Netzwerkprobleme bei allen Wiederholungsversuchen bestehen, erkennt Ihr Code `ClientException` Instanzen für Probleme wie Verbindungstimeouts, DNS-Auflösungsfehler und andere Verbindungsprobleme, die das SDK nicht lösen konnte.

Verwenden Sie das folgende Muster, um diese Ausnahmen in Ihrer Anwendung zu behandeln:

```
try {
    s3Client.getObject {
        bucket = "amzn-s3-demo-bucket"
        key = "my-key"
    }
} catch (e: AwsServiceException) {
    // Service-side errors that persisted through all retries.
    println("Service error after retries: ${e.errorDetails?.errorCode} - ${e.message}")

    // Handle specific service errors that couldn't be resolved.
    if (e.errorDetails?.errorCode == "ServiceQuotaExceededException" ||
        e.errorDetails?.errorCode == "ThrottlingException") {
        println("Rate limiting persisted - consider longer delays or quota increase")
    }
} catch (e: ClientException) {
    // Client-side errors (persistent network issues, DNS resolution failures, etc.)
    println("Client error after retries: ${e.message}")
}
```

Anpassen des Wiederholungsverhaltens

In den folgenden Abschnitten wird gezeigt, wie Sie das Wiederholungsverhalten des SDK an Ihren speziellen Anwendungsfall anpassen können.

Konfigurieren Sie die maximale Anzahl an Versuchen

Sie können die standardmäßige maximale Anzahl von Versuchen (3) im `retryStrategy`DSL-Block während der Client-Erstellung anpassen.

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

Mit dem im vorherigen Snippet gezeigten DynamoDB-Serviceclient versucht das SDK API-Aufrufe, die fehlschlagen, bis zu fünf Mal (der erste Versuch plus vier Wiederholungen).

Sie können automatische Wiederholungsversuche vollständig deaktivieren, indem Sie die maximale Anzahl von Versuchen auf einen festlegen, wie im folgenden Codeausschnitt gezeigt.

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

Konfigurieren Sie Verzögerungen und Backoff

Wenn ein Wiederholungsversuch erforderlich ist, wartet die standardmäßige Wiederholungsstrategie, bevor sie den nächsten Versuch durchführt. Die Verzögerung beim ersten Versuch ist gering, nimmt aber bei späteren Wiederholungen exponentiell zu. Die maximale Verzögerung ist begrenzt, sodass sie nicht zu groß wird.

Schließlich wird zufälliger Jitter auf die Verzögerungen zwischen allen Versuchen angewendet. Der Jitter trägt dazu bei, die Auswirkungen großer Flotten zu mildern, die zu erneuten Stürmen führen können. (Weitere Informationen zu exponentiellem Backoff und Jitter finden Sie in diesem AWS Architektur-Blogbeitrag.)

Die Verzögerungsparameter sind im DSL-Block konfigurierbar. `delayProvider`

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        delayProvider {
```

```
            initialDelay = 100.milliseconds
            maxBackoff = 5.seconds
        }
    }
}
```

Bei der im vorherigen Codeausschnitt gezeigten Konfiguration verzögert der Client den ersten Wiederholungsversuch um bis zu 100 Millisekunden. Die maximale Zeitspanne zwischen jedem Wiederholungsversuch beträgt 5 Sekunden.

Die folgenden Parameter sind für die Optimierung von Verzögerungen und Backoff verfügbar.

Parameter	Standardwert	Description
`initialDelay`	10 Millisekunden	Die maximale Verzögerung für den ersten Wiederholungsversuch. Wenn Jitter angewendet wird, kann die tatsächliche Verzögerung geringer sein.
`jitter`	1,0 (voller Jitter)	Die maximale Amplitude, um die die berechnete Verzögerung nach dem Zufallsprinzip reduziert werden soll. Der Standardwert 1,0 bedeutet, dass die berechnete Verzögerung auf einen beliebigen Wert von bis zu 100% reduziert werden kann (z. B. auf 0). Ein Wert von 0,5 bedeutet, dass die berechnete Verzögerung um bis zu die Hälfte reduziert werden kann. Somit könnte eine maximale Verzögerung von 10 ms auf einen Wert zwischen 5 ms und 10 ms reduziert werden. Ein

Parameter	Standardwert	Description
		Wert von 0,0 bedeutet, dass kein Jitter angewendet wird. **Important** Die Jitter-Konfiguration ist eine erweiterte Funktion. Eine Anpassung dieses Verhaltens wird normalerweise nicht empfohlen.
`maxBackoff`	20 Sekunden	Die maximale Verzögerung, die für jeden Versuch gelten soll. Wenn Sie diesen Wert festlegen, wird das exponentielle Wachstum zwischen aufeinanderfolgenden Versuchen begrenzt und verhindert, dass das berechnete Maximum zu groß ist. Dieser Parameter begrenzt die berechnete Verzögerung, bevor Jitter angewendet wird. Wenn Jitter angewendet wird, kann es sein, dass die Verzögerung noch weiter reduziert wird.

Parameter	Standardwert	Description
`scaleFactor`	1.5	Die exponentielle Basis, um die nachfolgende maximale Verzögerung erhöht wird. Bei einem Wert `initialDelay` von 10 ms und einem Wert `scaleFactor` von 1,5 würden beispielsweise die folgenden maximalen Verzögerungen berechnet: • Wiederholung 1:10 ms × 1,5° = 10 ms • Wiederholung 2:10 ms × 1,5¹ = 15 ms • Wiederholung 3:10 ms × 1,5² = 22,5 ms • Wiederholungsversuch 4:10 ms × 1,5³ = 33,75 ms Wenn Jitter angewendet wird, kann der tatsächliche Betrag jeder Verzögerung geringer sein.

Konfigurieren Sie den Token-Bucket für Wiederholungsversuche

Sie können das Verhalten der standardmäßigen Wiederholungsstrategie weiter ändern, indem Sie die Standardkonfiguration des Token-Buckets anpassen. Der Token-Bucket für Wiederholungen trägt dazu bei, Wiederholungsversuche zu reduzieren, bei denen die Erfolgswahrscheinlichkeit geringer ist oder deren Behebung mehr Zeit in Anspruch nehmen könnte, wie z. B. Timeout- und Drosselungsfehler.

> **Important**
>
> Die Token-Bucket-Konfiguration ist eine erweiterte Funktion. Das Anpassen dieses Verhaltens wird normalerweise nicht empfohlen.

Bei jedem erneuten Versuch (optional einschließlich des ersten Versuchs) wird die Kapazität des Token-Buckets um einen Teil verringert. Der dekrementierte Betrag hängt von der Art des Versuchs ab. Beispielsweise kann es billig sein, vorübergehende Fehler erneut zu versuchen, aber die Wiederholung von Timeout- oder Drosselungsfehlern kann teurer sein.

Ein erfolgreicher Versuch gibt dem Bucket wieder Kapazität zurück. Der Bucket darf nicht über seine maximale Kapazität hinaus erhöht und auch nicht unter Null reduziert werden.

Je nach Wert der `useCircuitBreakerMode` Einstellung führen Versuche, die Kapazität unter Null zu reduzieren, zu einem der folgenden Ergebnisse:

- Wenn die Einstellung TRUE ist, wird eine Ausnahme ausgelöst, z. B. wenn zu viele Wiederholungen stattgefunden haben und es unwahrscheinlich ist, dass weitere Versuche erfolgreich sind.
- Wenn die Einstellung FALSE ist, kommt es zu einer Verzögerung, z. B. zu Verzögerungen, bis der Bucket wieder ausreichend Kapazität hat.

> **Note**
>
> Wenn der Schutzschalter aktiviert wird (der Token-Bucket erreicht die Kapazität Null), gibt das SDK eine Meldung `ClientException` mit der Meldung „Kapazität erneut versuchen überschritten“ aus. Dies ist eine clientseitige Ausnahme, keine`AwsServiceException`, da sie eher auf die Wiederholungslogik des SDK als auf den Service zurückzuführen ist. AWS Die Ausnahme wird sofort ausgelöst, ohne dass der Vorgang versucht wird, wodurch Wiederholungsversuche bei Dienstausfällen vermieden werden.

Die Token-Bucket-Parameter sind im `tokenBucket`DSL-Block konfigurierbar:

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        tokenBucket {
```

```
            maxCapacity = 100
            refillUnitsPerSecond = 2
        }
    }
}
```

Die folgenden Parameter sind für die Optimierung des Retry-Token-Buckets verfügbar:

Parameter	Standardwert	Description
`initialTryCost`	0	Der Betrag, der bei ersten Versuchen aus dem Bucket herabgesetzt werden soll. Der Standardwert 0 bedeutet, dass keine Kapazität verringert wird und somit die ersten Versuche nicht gestoppt oder verzögert werden.
`initialTrySuccessIncrement`	1	Der Betrag, um den die Kapazität erhöht werden soll, wenn der erste Versuch erfolgreich war.
`maxCapacity`	500	Die maximale Kapazität des Token-Buckets. Die Anzahl der verfügbaren Token darf diese Anzahl nicht überschreiten.
`refillUnitsPerSecond`	0	Die Menge an Kapazität, die dem Bucket jede Sekunde wieder hinzugefügt wird. Ein Wert von 0 bedeutet, dass keine Kapazität automatisch wieder hinzugefügt wird. (Beispielsweise führen nur erfolgreiche Versuche zu einer

Parameter	Standardwert	Description
		Erhöhung der Kapazität). Ein Wert von 0 `useCircuitBreakerMode` muss TRUE sein.
`retryCost`	5	Der Betrag, der bei einem Versuch nach einem vorübergehenden Ausfall aus dem Bucket herabgesetzt werden soll. Wenn der Versuch erfolgreich ist, wird derselbe Betrag wieder zurück in den Bucket inkrementiert.
`timeoutRetryCost`	10	Der Betrag, der bei einem Versuch nach einem Timeout- oder Drosselungsfehler vom Bucket heruntergezählt werden soll. Wenn der Versuch erfolgreich ist, wird derselbe Betrag wieder auf den Bucket erhöht.

Parameter	Standardwert	Description
`useCircuitBreakerMode`	TRUE	Bestimmt das Verhalten, wenn ein Versuch, die Kapazität zu verringern, dazu führen würde, dass die Kapazität des Buckets unter Null fällt. Bei TRUE löst der Token-Bucket eine Ausnahme aus, die angibt, dass keine Wiederholungskapazität mehr vorhanden ist. Bei FALSE verzögert der Token-Bucket den Versuch, bis genügend Kapazität wieder aufgefüllt ist.

Ausführliche Informationen zu Ausnahmetypen, die bei Wiederholungsszenarien ausgelöst werden, einschließlich Ausnahmen bei Leistungsschaltern, finden Sie unter. the section called “Welche Ausnahmen erreichen Ihren Code, wenn Wiederholungsversuche fehlschlagen”

Konfigurieren Sie adaptive Wiederholungsversuche

Als Alternative zur standardmäßigen Wiederholungsstrategie ist die adaptive Wiederholungsstrategie ein fortschrittlicher Ansatz, bei dem nach der idealen Anforderungsrate gesucht wird, um Drosselungsfehler zu minimieren.

> **Important**
>
> Adaptive Wiederholungen sind ein erweiterter Wiederholungsmodus. Die Verwendung dieser Wiederholungsstrategie wird normalerweise nicht empfohlen.

Adaptive Wiederholungen umfassen alle Funktionen von Standardwiederholungen. Es fügt einen clientseitigen Ratenbegrenzer hinzu, der die Rate gedrosselter Anfragen im Vergleich zu Anfragen ohne Drosselung misst. Außerdem wird der Datenverkehr so begrenzt, dass versucht wird, innerhalb einer sicheren Bandbreite zu bleiben, sodass im Idealfall keine Drosselungsfehler auftreten.

Die Rate passt sich in Echtzeit an sich ändernde Betriebsbedingungen und Verkehrsmuster an und kann die Verkehrsrate entsprechend erhöhen oder verringern. Entscheidend ist, dass der Ratenbegrenzer erste Versuche in Szenarien mit hohem Verkehrsaufkommen verzögern kann.

Sie wählen die adaptive Wiederholungsstrategie aus, indem Sie der Methode einen zusätzlichen Parameter hinzufügen. `retryStrategy` Die Parameter des Ratenbegrenzers sind im `rateLimiter`DSL-Block konfigurierbar.

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

> **Note**
>
> Bei der adaptiven Wiederholungsstrategie wird davon ausgegangen, dass der Client mit einer einzelnen Ressource arbeitet (z. B. einer DynamoDB-Tabelle oder einem Amazon S3 S3-Bucket).
> Wenn Sie einen einzelnen Client für mehrere Ressourcen verwenden, führen Drosselungen oder Ausfälle im Zusammenhang mit einer Ressource zu einer erhöhten Latenz und zu Ausfällen, wenn der Client auf alle anderen Ressourcen zugreift. Wenn Sie die Strategie der adaptiven Wiederholung verwenden, empfehlen wir, für jede Ressource einen einzelnen Client zu verwenden.

Beobachtbarkeit

Beobachtbarkeit ist das Ausmaß, in dem der aktuelle Zustand eines Systems aus den von ihm ausgegebenen Daten abgeleitet werden kann. Die ausgegebenen Daten werden allgemein als Telemetrie bezeichnet.

AWS SDK für Kotlin Sie können alle drei gängigen Telemetriesignale bereitstellen: Metriken, Traces und Logs. Sie können eine Verbindung herstellen `TelemetryProvider`, um Telemetriedaten an ein

Observability-Backend (z. B. AWS X-Rayoder Amazon CloudWatch) zu senden und dann darauf zu reagieren.

Standardmäßig ist nur die Protokollierung aktiviert und andere Telemetriesignale sind im SDK deaktiviert. In diesem Thema wird erklärt, wie die Telemetrieausgabe aktiviert und konfiguriert wird.

> **Important**
>
> `TelemetryProvider`ist derzeit eine experimentelle API, deren Verwendung aktiviert werden muss.

Konfigurieren Sie ein `TelemetryProvider`

Sie können a `TelemetryProvider` in Ihrer Anwendung global für alle Service-Clients oder für einzelne Clients konfigurieren. In den folgenden Beispielen wird eine hypothetische `getConfiguredProvider()` Funktion verwendet, um die `TelemetryProvider` API-Operationen zu demonstrieren. In the section called “Anbieter von Telemetriedaten” diesem Abschnitt werden Informationen zu Implementierungen beschrieben, die vom SDK bereitgestellt werden. Wenn ein Anbieter nicht unterstützt wird, können Sie Ihren eigenen Support implementieren oder eine Funktionsanfrage stellen. GitHub

Konfigurieren Sie den globalen Standardtelemetrie-Anbieter

Standardmäßig versucht jeder Dienstclient, den weltweit verfügbaren Telemetrieanbieter zu verwenden. Auf diese Weise können Sie den Anbieter einmal festlegen, und alle Clients verwenden ihn. Dies sollte nur einmal durchgeführt werden, bevor Sie Service-Clients instanziieren.

Um den globalen Telemetrieanbieter zu verwenden, aktualisieren Sie zunächst Ihre Projektabhängigkeiten, um das Telemetriestandardmodul hinzuzufügen, wie im folgenden Gradle-Snippet gezeigt.

(Sie können zum *`X.Y.Z`* Link navigieren, um die neueste verfügbare Version zu sehen.)

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

Stellen Sie dann den globalen Telemetrieanbieter ein, bevor Sie einen Service-Client erstellen, wie im folgenden Code gezeigt.

```
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")
}
```

Konfigurieren Sie einen Telemetrieanbieter für einen bestimmten Dienstclient

Sie können einen einzelnen Dienstclient mit einem bestimmten Telemetrieanbieter (mit Ausnahme des globalen) konfigurieren. Dies wird im folgenden Beispiel veranschaulicht.

```
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")
}
```

Kennzahlen

In der folgenden Tabelle sind die Telemetriemetriken aufgeführt, die das SDK ausgibt. Konfigurieren Sie einen Telemetrieanbieter, um die Messwerte beobachtbar zu machen.

Welche Metriken werden ausgegeben?

Metrikname	Einheiten	Typ	Attribute	Description
smithy.client.call.duration	S	Histogram	rpc.service, rpc.method	Gesamtdauer des Anrufs (einschließlich Wiederholungen)
smithy.client.call.attempts	{Versuch}	Monoton Counter	rpc.service, rpc.method	Die Anzahl der Versuche für einen einzelnen Vorgang
smithy.client.call.errors	{Fehler}	Monoton Counter	rpc.service, rpc.method, exception.type	Die Anzahl der Fehler bei einem Vorgang
smithy.client.call.attempt_duration	S	Histogram	rpc.service, rpc.method	Die Zeit, die benötigt wird, um eine Verbindung mit dem Dienst herzustellen, die Anfrage zu senden und den HTTP-Statuscode und die Header zurückzubekommen (einschließlich der Wartezeit in der Warteschlange auf das Senden)
smithy.client.call.resolve_endpoint_duration	S	Histogram	rpc.service, rpc.method	Die Zeit, die benötigt wird, um einen Endpunkt (Endpoint Resolver, nicht DNS) für die Anfrage aufzulösen
smithy.client.call.serialization_duration	S	Histogram	rpc.service, rpc.method	Die Zeit, die benötigt wird, um einen Nachrichtentext zu serialisieren
smithy.client.call.deserialization_duration	S	Histogram	rpc.service, rpc.method	Die Zeit, die benötigt wird, um einen Nachrichtentext zu deserialisieren

Metrikname	Einheiten	Typ	Attribute	Description
smithy.client.call.auth.signing_duration	S	Histogram	rpc.service, rpc.method, auth.scheme_id	Die Zeit, die benötigt wird, um eine Anfrage zu signieren
smithy.client.call.auth.resolve_identity_duration	S	Histogram	rpc.service, rpc.method, auth.scheme_id	Die Zeit, die benötigt wird, um eine Identität (wie Anmeldeinformationen oder ein Bearer-Token) von einem Identitätsanbieter abzurufen AWS
smithy.client.http.connections.acquire_duration	S	Histogram		Die Zeit, die eine Anfrage benötigt, um eine Verbindung herzustellen
smithy.client.http.connections.limit	{Verbindung}	[Asynchron] UpDownCounter		Die maximale Anzahl offener Verbindungen allowed/configured für den HTTP-Client
smithy.client.http.connections.usage	{Verbindung}	[Asynchron] UpDownCounter	Zustand: inaktiv | erworben	Aktueller Status des Verbindungspools
smithy.client.http.connections.uptime	S	Histogram		Die Zeit, in der eine Verbindung geöffnet war
smithy.client.http.requests.usage	{Anfrage}	[Asynchron] UpDownCounter	Status: in der Warteschlange | während des Fluges	Der aktuelle Status der Parallelität von HTTP-Client-Anfragen

Metrikname	Einheiten	Typ	Attribute	Description
smithy.client.http.requests.queued_duration	S	Histogram		Die Zeit, die eine Anfrage in der Warteschlange verbracht hat und darauf gewartet hat, vom HTTP-Client ausgeführt zu werden
smithy.client.http.bytes_sent	Von	MonotonCounter	server.adresse	Die Gesamtzahl der vom HTTP-Client gesendeten Byte
smithy.client.http.bytes_received	Von	MonotonCounter	server.adresse	Die Gesamtzahl der vom HTTP-Client empfangenen Byte

Im Folgenden finden Sie die Spaltenbeschreibungen:

- Metrikname — Der Name der ausgegebenen Metrik.
- Einheiten — Die Maßeinheit für die Metrik. Die Einheiten werden in der UCUM-Notation mit Berücksichtigung der Groß- und Kleinschreibung („c/s“) angegeben.
- Typ — Die Art des Instruments, das zur Erfassung der Metrik verwendet wird.
- Beschreibung — Eine Beschreibung dessen, was mit der Metrik gemessen wird.
- Attribute — Der Satz von Attributen (Dimensionen), die mit der Metrik ausgegeben werden.

Protokollierung

Der AWS SDK für Kotlin konfiguriert einen SLF4J-kompatiblen Logger als Standard `LoggerProvider` des Telemetrieanbieters. Mit SLF4 J, einer Abstraktionsschicht, können Sie zur Laufzeit eines von mehreren Protokollierungssystemen verwenden. Zu den unterstützten Protokollierungssystemen gehören Java Logging APIs, Log4J 2 und Logback.

> **Warning**
>
> Wir empfehlen, Wire Logging nur für Debugging-Zwecke zu verwenden. (Die Kabelprotokollierung wird weiter unten beschrieben.) Schalten Sie es in Ihren Produktionsumgebungen aus, da es sensible Daten wie E-Mail-Adressen, Sicherheitstoken, API-Schlüssel, Passwörter und AWS Secrets Manager Geheimnisse protokollieren kann.

Wire Logging protokolliert die gesamte Anfrage oder Antwort ohne Verschlüsselung, selbst bei einem HTTPS-Anruf.
Bei großen Anfragen (wie dem Hochladen einer Datei auf Amazon S3) oder Antworten kann die ausführliche Kabelprotokollierung auch die Leistung Ihrer Anwendung erheblich beeinträchtigen.

Beispiel für eine Log4j 2-Logging-Konfiguration

Obwohl jede `SLF4J` -kompatible Protokollbibliothek verwendet werden kann, ermöglicht dieses Beispiel die Protokollausgabe aus dem SDK in JVM-Programmen, die Log4j 2 verwenden:

Gradle-Abhängigkeiten

(Sie können zum *`X.Y.Z`* Link navigieren, um die neueste verfügbare Version zu sehen.)

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

Log4j 2-Konfigurationsdatei

Erstellen Sie eine Datei mit dem Namen `log4j2.xml` in Ihrem `resources` Verzeichnis (zum Beispiel`<project-dir>/src/main/resources`). Fügen Sie der Datei die folgende XML-Konfiguration hinzu:

```
<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>
```

Diese Konfiguration enthält den `%X` Spezifizierer in dem `pattern` Attribut, das die MDC-Protokollierung (Mapped Diagnostic Context) aktiviert.

Das SDK fügt für jeden Vorgang die folgenden MDC-Elemente hinzu.

rpc

Zum Beispiel der Name des aufgerufenen RPC. `S3.GetObject`

sdkInvocationId

Eine eindeutige ID, die vom Service-Client für den Vorgang zugewiesen wurde. Die ID korreliert alle Protokollierungsereignisse, die sich auf den Aufruf eines einzelnen Vorgangs beziehen.

Geben Sie den Protokollmodus für Nachrichten auf Kabelebene an

Standardmäßig protokolliert der AWS SDK für Kotlin keine Nachrichten auf drahtgebundener Ebene, da sie vertrauliche Daten aus API-Anfragen und -Antworten enthalten können. Manchmal benötigen Sie diese Detailgenauigkeit jedoch für Debugging-Zwecke.

Mit dem Kotlin SDK können Sie im Code oder mithilfe von Umgebungseinstellungen einen Protokollmodus einrichten, um Debug-Meldungen für Folgendes zu aktivieren:

- HTTP-Anforderungen
- HTTP-Antworten

Der Protokollmodus wird durch ein Bitfeld unterstützt, in dem jedes Bit ein Flag (Modus) ist und die Werte additiv sind. Sie können einen Anforderungsmodus und einen Antwortmodus kombinieren.

Stellen Sie den Protokollmodus im Code ein

Um zusätzliche Protokollierung zu aktivieren, legen Sie die `logMode` Eigenschaft fest, wenn Sie einen Service-Client erstellen.

Das folgende Beispiel zeigt, wie die Protokollierung von Anfragen (mit dem Hauptteil) und der Antwort (ohne Hauptteil) aktiviert wird.

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

// ...

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

```
}
```

Ein Wert für den Protokollmodus, der während der Erstellung des Service-Clients festgelegt wurde, hat Vorrang vor allen in der Umgebung festgelegten Protokollmodus-Werten.

Legt den Protokollmodus in der Umgebung fest

Verwenden Sie eine der folgenden Methoden, um global einen Protokollmodus für alle Service-Clients festzulegen, die nicht explizit im Code konfiguriert sind:

- JVM-Systemeigenschaft: `sdk.logMode`
- Umgebungsvariable: `SDK_LOG_MODE`

Die folgenden Werte sind verfügbar, bei denen Groß- und Kleinschreibung nicht beachtet wird:

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

Um einen kombinierten Protokollmodus mit Einstellungen aus der Umgebung zu erstellen, trennen Sie die Werte durch ein Pipe-Symbol (`|`).

In den folgenden Beispielen wird beispielsweise derselbe Protokollmodus wie im vorherigen Beispiel festgelegt.

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

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

> **Note**
>
> Sie müssen außerdem einen kompatiblen SLF4 J-Logger konfigurieren und die Protokollierungsebene auf DEBUG setzen, um die Protokollierung auf Kabelebene zu aktivieren.

Anbieter von Telemetriedaten

Das SDK unterstützt derzeit OpenTelemetry(OTel) als Anbieter. Das SDK könnte in future weitere Telemetrieanbieter anbieten.

Themen

- Konfigurieren Sie den OpenTelemetry basierten Telemetrieanbieter

Konfigurieren Sie den OpenTelemetry basierten Telemetrieanbieter

Das SDK für Kotlin bietet eine Implementierung der `TelemetryProvider` Schnittstelle, die von OpenTelemetry unterstützt wird.

Voraussetzungen

Aktualisieren Sie Ihre Projektabhängigkeiten, um den OpenTelemetry Anbieter hinzuzufügen, wie im folgenden Gradle-Snippet gezeigt. Sie können zum *X.Y.Z* Link navigieren, um die neueste verfügbare Version zu sehen.

```
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")
}
```

Das SDKs konfigurieren

Mit dem folgenden Code wird ein Dienstclient mithilfe des OpenTelemetry Telemetrieanbieters konfiguriert.

```
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**
>
> Eine Erläuterung der Konfiguration des OpenTelemetry SDK würde den Rahmen dieses Handbuchs sprengen. Die OpenTelemetryJava-Dokumentation enthält Konfigurationsinformationen zu den verschiedenen Ansätzen: manuell, automatisch über den Java-Agenten oder den (optionalen) Collector.

Ressourcen

Die folgenden Ressourcen stehen Ihnen für den Einstieg zur Verfügung OpenTelemetry.

- AWS Distro für OpenTelemetry - AWS OTe L Distro-Homepage
- aws-otel-java-instrumentation- AWS Instrumentierungsbibliothek für Distro für Java OpenTelemetry
- aws-otel-lambda- AWS verwaltete OpenTelemetry Lambda-Schichten
- aws-otel-collector- AWS Distribution für Collector OpenTelemetry
- AWS Bewährte Methoden zur Beobachtbarkeit — Allgemeine bewährte Verfahren für Beobachtbarkeit, spezifisch für AWS

Überschreiben Sie die Service-Client-Konfiguration

Nachdem ein Service-Client erstellt wurde, verwendet der Service-Client eine feste Konfiguration für alle Operationen. Manchmal müssen Sie jedoch möglicherweise die Konfiguration für einen oder mehrere bestimmte Operationen überschreiben.

Jeder Service-Client verfügt über eine `withConfig` Erweiterung, sodass Sie eine Kopie der vorhandenen Konfiguration ändern können. Die `withConfig` Erweiterung gibt einen neuen Service-Client mit einer geänderten Konfiguration zurück. Der ursprüngliche Client existiert unabhängig und verwendet seine ursprüngliche Konfiguration.

Das folgende Beispiel zeigt die Erstellung einer `S3Client` Instanz, die zwei Operationen aufruft.

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

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

Der folgende Ausschnitt zeigt, wie die Konfiguration für einen einzelnen `listObjectV2` Vorgang überschrieben wird.

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

Die Operationsaufrufe auf dem `s3` Client verwenden die ursprüngliche Konfiguration, die bei der Erstellung des Clients angegeben wurde. Die Konfiguration umfasst die Protokollierung von Anfragen und `us-west-2 region` für die Region.

Der `listObjectsV2` Aufruf auf dem `overriddenS3` Client verwendet dieselben Einstellungen wie auf dem ursprünglichen `s3` Client, mit Ausnahme der Region, die jetzt `eu-central-1` verfügbar ist.

Lebenszyklus eines überschriebenen Clients

Im vorherigen Beispiel sind der `s3` Client und der `overriddenS3` Client unabhängig voneinander. Operationen können auf beiden Clients aufgerufen werden, solange sie geöffnet sind. Jeder verwendet eine separate Konfiguration, sie können jedoch die zugrunde liegenden Ressourcen (z. B. eine HTTP-Engine) gemeinsam nutzen, sofern diese nicht ebenfalls überschrieben werden.

Sie schließen einen Client mit einer überschriebenen Konfiguration und den ursprünglichen Client getrennt. Sie können einen Client mit überschriebener Konfiguration vor oder nach dem Schließen seines ursprünglichen Clients schließen. Sofern Sie einen Client mit überschriebener Konfiguration nicht über einen längeren Zeitraum verwenden müssen, empfehlen wir, seinen Lebenszyklus mit dieser Methode abzuschließen. `use` Die `use` Methode stellt sicher, dass der Client geschlossen wird, falls Ausnahmen auftreten.

Ressourcen, die von Clients gemeinsam genutzt werden

Wenn Sie einen Service-Client mithilfe von verwenden`withConfig`, teilt er sich möglicherweise Ressourcen mit dem ursprünglichen Client. Wenn Sie dagegen einen Client mithilfe von FromEnvironment erstellen oder ihn explizit konfigurieren, verwendet der Client unabhängige Ressourcen. Ressourcen wie HTTP-Engines und Anbieter von Anmeldeinformationen werden gemeinsam genutzt, sofern sie nicht im Block überschrieben werden. `withConfig`

Da der Lebenszyklus jedes Clients unabhängig ist, bleiben gemeinsam genutzte Ressourcen geöffnet und nutzbar, bis der letzte Client geschlossen wird. Daher ist es wichtig, dass Sie Service-Clients schließen, die überschrieben wurden, wenn Sie sie nicht mehr benötigen. Dadurch wird verhindert, dass gemeinsam genutzte Ressourcen offen bleiben und Systemressourcen wie Speicher-, Verbindungs- und CPU-Zyklen verbrauchen.

Das folgende Beispiel zeigt sowohl gemeinsam genutzte als auch unabhängige Ressourcen.

Die `overriddenS3` Clients `s3` und die Clients verwenden dieselbe Credentials Provider-Instanz, einschließlich ihrer Caching-Konfiguration. Aufrufe, die mit `overriddenS3` Reuse Credentials getätigt wurden, wenn der zwischengespeicherte Wert noch aktuell ist und aus Aufrufen des `s3` Clients stammt.

Die HTTP-Engine wird nicht von den beiden Clients gemeinsam genutzt. Jeder Client hat eine unabhängige HTTP-Engine, da diese beim Aufruf außer Kraft gesetzt wurde. `withConfig`

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

s3.listBuckets { ... }

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

Verwenden Sie das SDK

Dieser Abschnitt enthält grundlegende Informationen, die für die Verwendung von erforderlich sind AWS SDK für Kotlin.

Themen

- Anfragen stellen
- Coroutinen
- Streaming-Operationen
- Paginierung
- Waiter
- Fehlerbehandlung
- Anfragen vorab signieren
- Problembehebung FAQs
- Spott in der AWS SDK für Kotlin

Anfragen stellen

Verwenden Sie einen Service-Client, um Anfragen an einen zu stellen AWS-Service. Der AWS SDK für Kotlin stellt domänenspezifische Sprachen (DSLs) zur Verfügung, die einem typsicheren Builder-Muster folgen, um Anfragen zu erstellen. Auf verschachtelte Strukturen von Anfragen kann auch über sie zugegriffen werden. DSLs

Das folgende Beispiel zeigt, wie Sie eine Amazon DynamoDB CreateTable-Operationseingabe erstellen:

```
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-Überladungen der Serviceschnittstelle

Jeder Nicht-Streaming-Vorgang auf der Service-Client-Schnittstelle weist eine DSL-Überlastung auf, sodass Sie keine separate Anforderung erstellen müssen.

Beispiel für die Erstellung eines Amazon Simple Storage Service (Amazon S3) -Buckets mit der überladenen Funktion:

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

Dies entspricht:

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

```
}

s3client.createBucket(request)
```

Anfragen ohne erforderliche Eingaben

Operationen, für die keine erforderlichen Eingaben erforderlich sind, können aufgerufen werden, ohne dass ein Anforderungsobjekt übergeben werden muss. Dies ist häufig bei Operationen vom Typ Liste möglich, z. B. bei der Amazon `listBuckets` S3-API-Operation.

Die folgenden drei Anweisungen sind beispielsweise gleichwertig:

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

Coroutinen

Der AWS SDK für Kotlin ist standardmäßig asynchron. Das SDK für Kotlin verwendet `suspend` Funktionen für alle Operationen, die dazu bestimmt sind, von einer Coroutine aus aufgerufen zu werden.

Eine ausführlichere Anleitung zu Coroutinen finden Sie in der offiziellen Kotlin-Dokumentation.

Gleichzeitige Anfragen stellen

Der asynchrone Coroutine Builder kann verwendet werden, um gleichzeitige Anfragen zu starten, bei denen Ihnen die Ergebnisse wichtig sind. `async`gibt einen Wert vom Typ Deferred zurück, was für eine leichtgewichtige, nicht blockierende future steht, die ein Versprechen darstellt, zu einem späteren Zeitpunkt ein Ergebnis zu liefern.

Wenn Ihnen die Ergebnisse egal sind (nur, dass ein Vorgang abgeschlossen wurde), können Sie den Launch Coroutine Builder verwenden. `launch`ist konzeptionell ähnlich wie. `async` Der Unterschied besteht darin, dass launch einen Job zurückgibt und keinen Ergebniswert enthält, während launch a `async` zurückgibt`Deferred`.

Im Folgenden finden Sie ein Beispiel für gleichzeitige Anfragen an Amazon S3 mithilfe der Operation headObject, um die Inhaltsgröße von zwei Schlüsseln abzurufen:

```
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")

}
```

Blockierungsanfragen stellen

Um Serviceaufrufe von vorhandenem Code aus zu tätigen, der keine Coroutinen verwendet und ein anderes Threading-Modell implementiert, können Sie den RunBlocking-Coroutine-Builder verwenden. Ein Beispiel für ein anderes Threading-Modell ist die Verwendung des traditionellen Ansatzes von Java. executors/futures Möglicherweise müssen Sie diesen Ansatz verwenden, wenn Sie Java- und Kotlin-Code oder -Bibliotheken mischen.

Wie der Name schon sagt, startet dieser `runBlocking` Builder eine neue Coroutine und blockiert den aktuellen Thread, bis er abgeschlossen ist.

> **Warning**
>
> `runBlocking`sollte im Allgemeinen nicht von einer Coroutine aus verwendet werden. Es wurde entwickelt, um regulären Blockierungscode mit Bibliotheken zu verbinden, die im Suspending-Stil geschrieben sind (z. B. in Hauptfunktionen und Tests).

Streaming-Operationen

In der AWS SDK für Kotlin werden Binärdaten (Streams) als `ByteStream`Typ dargestellt, bei dem es sich um einen abstrakten, schreibgeschützten Bytestrom handelt.

Streaming-Antworten

Antworten mit einem binären Stream (z. B. der GetObjectAPI-Vorgang Amazon Simple Storage Service (Amazon S3)) werden anders behandelt als andere Methoden. Diese Methoden verwenden eine Lambda-Funktion, die die Antwort verarbeitet, anstatt die Antwort direkt zurückzugeben. Dadurch wird der Umfang der Antwort auf die Funktion beschränkt und die Lebenszeitverwaltung sowohl für den Aufrufer als auch für die SDK-Laufzeit vereinfacht.

Nach der Rückkehr der Lambda-Funktion werden alle Ressourcen wie die zugrunde liegende HTTP-Verbindung freigegeben. (Auf sie `ByteStream` sollte nicht zugegriffen werden, nachdem das Lambda zurückgekehrt ist, und sie sollten auch nicht nach der Schließung weitergegeben werden.) Das Ergebnis des Aufrufs ist das, was das Lambda zurückgibt.

Das folgende Codebeispiel zeigt, wie die Funktion GetObject einen Lambda-Parameter empfängt, der die Antwort verarbeitet.

```
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")
```

Der `ByteStream` Typ hat die folgenden Erweiterungen für gängige Verwendungsarten:

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

All dies ist im `aws.smithy.kotlin.runtime.content` Paket definiert.

Streaming-Anfragen

Um eine bereitzustellen`ByteStream`, gibt es auch mehrere praktische Methoden, darunter die folgenden:

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

All dies ist im `aws.smithy.kotlin.runtime.content` Paket definiert.

Das folgende Codebeispiel zeigt die Verwendung `ByteStream` praktischer Methoden, die die Eigenschaft body bei der Erstellung eines bereitstellen PutObjectRequest:

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

Paginierung

Viele AWS Operationen geben paginierte Ergebnisse zurück, wenn die Nutzlast zu groß ist, um sie in einer einzigen Antwort zurückzugeben. AWS SDK für Kotlin Dazu gehören Erweiterungen der Service-Client-Schnittstelle, die die Ergebnisse auto für Sie paginieren. Sie müssen nur den Code schreiben, der die Ergebnisse verarbeitet.

Die Paginierung wird als Flow bereitgestellt<T>, sodass Sie die idiomatischen Transformationen von Kotlin für asynchrone Sammlungen (wie`map`, und) nutzen können. `filter` `take` Ausnahmen sind transparent, sodass sich die Fehlerbehandlung wie ein regulärer API-Aufruf anfühlt, und die Stornierung entspricht der allgemeinen kooperativen Annullierung von Coroutinen. Weitere Informationen finden Sie im offiziellen Leitfaden unter Abläufe und Flow-Ausnahmen.

> **Note**
>
> In den folgenden Beispielen wird Amazon S3 verwendet. Die Konzepte sind jedoch für jeden Service identisch, der über einen oder mehrere Seiten verfügt. APIs Alle Paginierungserweiterungen sind im `aws.sdk.kotlin.services.<service>.paginators` Paket definiert (z. B.`aws.sdk.kotlin.services.dynamodb.paginators`).

Das folgende Codebeispiel zeigt, wie Sie die paginierte Antwort aus dem Funktionsaufruf ListObjectsV2Paginated verarbeiten können.

Importe

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

Code

```
val s3 = S3Client.fromEnvironment()
val req = ListObjectsV2Request {
    bucket = "amzn-s3-demo-bucket"
    maxKeys = 1
}

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

Waiter

Waiter sind eine clientseitige Abstraktion, die verwendet wird, um eine Ressource abzufragen, bis ein gewünschter Status erreicht ist oder bis festgestellt wird, dass die Ressource nicht in den gewünschten Zustand übergeht. Dies ist eine häufige Aufgabe, wenn Sie mit Diensten arbeiten, die irgendwann konsistent sind, wie Amazon Simple Storage Service (Amazon S3), oder mit Diensten, die asynchron Ressourcen erstellen, wie Amazon EC2.

Das Schreiben von Logik zur kontinuierlichen Abfrage des Status einer Ressource kann umständlich und fehleranfällig sein. Das Ziel der Kellner ist es, diese Verantwortung aus dem Kundencode in den Kundencode zu verlagern AWS SDK für Kotlin, der über fundierte Kenntnisse der zeitlichen Abläufe verfügt. AWS

> **Note**
>
> In den folgenden Beispielen wird Amazon S3 verwendet. Die Konzepte sind jedoch für alle AWS-Service , für die ein oder mehrere Kellner definiert sind, dieselben. Alle Erweiterungen sind im `aws.sdk.kotlin.services.<service>.waiters` Paket definiert (z. B.`aws.sdk.kotlin.services.dynamodb.waiters`). Sie folgen außerdem einer Standardbenennungskonvention (`waitUntil<Condition>`).

Das folgende Codebeispiel zeigt die Verwendung einer Kellnerfunktion, mit der Sie das Schreiben von Abfragelogik vermeiden können.

Importe

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

Code

```
val s3 = S3Client.fromEnvironment()

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

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

// The bucket now exists.
```

> **Note**
>
> Jede Wartemethode gibt eine `Outcome` Instanz zurück, die verwendet werden kann, um die endgültige Antwort zu erhalten, die dem Erreichen der gewünschten Bedingung entspricht. Ein Ergebnis enthält auch zusätzliche Details wie die Anzahl der Versuche, den gewünschten Status zu erreichen.

Fehlerbehandlung

Es ist wichtig zu verstehen, wie AWS SDK für Kotlin und wann Ausnahmen ausgelöst werden, um qualitativ hochwertige Anwendungen mithilfe des SDK zu erstellen. In den folgenden Abschnitten werden die verschiedenen Fälle von Ausnahmen beschrieben, die vom SDK ausgelöst werden, und wie sie korrekt verarbeitet werden.

Ausnahmen für Dienste

Die häufigste Ausnahme ist die`AwsServiceException`, von der alle dienstspezifischen Ausnahmen (wie`S3Exception`) erben. Diese Ausnahme stellt eine Fehlerantwort von einem dar. AWS-Service Wenn Sie beispielsweise versuchen, eine EC2 Amazon-Instance zu beenden, die nicht existiert, gibt Amazon eine Fehlerantwort EC2 zurück. Die Details zur Fehlerantwort sind in der Datei enthalten`AwsServiceException`, die ausgelöst wird.

Wenn Sie auf eine stoßen`AwsServiceException`, bedeutet dies, dass Ihre Anfrage erfolgreich an die gesendet wurde, AWS-Service aber nicht bearbeitet werden konnte. Dies kann an Fehlern in den Parametern der Anforderung oder an Problemen auf Seiten des Services liegen.

Ausnahmen für Kunden

`ClientException`weist darauf hin, dass im AWS SDK für Kotlin Client-Code ein Problem aufgetreten ist, entweder beim Versuch, eine Anfrage an zu senden, AWS oder beim Versuch, eine Antwort von AWS zu analysieren. A `ClientException` ist im Allgemeinen schwerwiegender als ein `AwsServiceException` und weist darauf hin, dass ein schwerwiegendes Problem darin besteht, den Client daran zu hindern, Serviceanrufe an zu AWS-Services verarbeiten. Beispielsweise gibt der Befehl AWS SDK für Kotlin a aus, `ClientException` wenn er eine Antwort von einem Dienst nicht analysieren kann.

Fehler-Metadaten

Jede Service- und Client-Ausnahme hat die `sdkErrorMetadata` Eigenschaft. Dies ist ein typisierter Eigenschaften-Bag, der verwendet werden kann, um zusätzliche Informationen zu dem Fehler abzurufen.

Für den `AwsErrorMetadata` Typ existieren direkt mehrere vordefinierte Erweiterungen, einschließlich, aber nicht beschränkt auf die folgenden:

- `sdkErrorMetadata.requestId`— die eindeutige Anfrage-ID
- `sdkErrorMetadata.errorMessage`— die für Menschen lesbare Nachricht (entspricht normalerweise der`Exception.message`, kann aber mehr Informationen enthalten, wenn die Ausnahme dem Dienst unbekannt war)
- `sdkErrorMetadata.protocolResponse`— Die rohe Protokollantwort

Das folgende Beispiel zeigt den Zugriff auf die Fehlermetadaten.

```
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}")
}
```

Anfragen vorab signieren

Sie können Anfragen für einige AWS API-Operationen vorab signieren, sodass ein anderer Aufrufer die Anfrage später verwenden kann, ohne seine eigenen Anmeldeinformationen angeben zu müssen.

Nehmen wir zum Beispiel an, dass Alice Zugriff auf ein Amazon Simple Storage Service (Amazon S3) -Objekt hat und den Objektzugriff vorübergehend mit Bob teilen möchte. Alice kann eine vorab signierte `GetObject` Anfrage zur Weitergabe an Bob generieren, sodass er das Objekt herunterladen kann, ohne Zugriff auf Alices Anmeldeinformationen zu benötigen.

Grundlagen der Vorsignierung

Das SDK für Kotlin bietet Erweiterungsmethoden für Service-Clients zur Vorsignierung von Anfragen. Alle vorsignierten Anfragen benötigen eine Dauer, die angibt, wie lange die signierte Anfrage gültig ist. Nach Ablauf der Dauer läuft die vorsignierte Anforderung ab und löst einen Authentifizierungsfehler aus, wenn sie ausgeführt wird.

Der folgende Code zeigt ein Beispiel, das eine vorsignierte `GetObject` Anfrage für Amazon S3 erstellt. Die Anfrage ist nach ihrer Erstellung 24 Stunden lang gültig.

```
val s3 = S3Client.fromEnvironment()

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

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

Die `presignGetObject`Erweiterungsmethode gibt ein `HttpRequest`Objekt zurück. Das Anforderungsobjekt enthält die vorsignierte URL, über die die Operation aufgerufen werden kann. Ein anderer Aufrufer kann die URL (oder die gesamte Anfrage) in einer anderen Codebasis- oder Programmiersprachenumgebung verwenden.

Nachdem Sie die vorsignierte Anfrage erstellt haben, verwenden Sie einen HTTP-Client, um eine Anfrage aufzurufen. Die API zum Aufrufen einer HTTP-GET-Anfrage hängt vom HTTP-Client ab. Das folgende Beispiel verwendet die Kotlin-Methode `URL.readText`.

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

Erweiterte Presigning-Konfiguration

Im SDK verfügt jede Methode, mit der Anfragen vorab signiert werden können, über eine Vielzahl von Funktionen, die Sie verwenden können, um erweiterte Konfigurationsoptionen bereitzustellen, z. B. eine spezifische Implementierung für den Unterzeichner oder detaillierte Signaturparameter.

Das folgende Beispiel zeigt eine Amazon S3 `GetObject` S3-Anfrage, die die CRT-Signaturvariante verwendet und ein future Signaturdatum angibt.

```
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
}
```

Die zurückgesandte, vorsignierte Anfrage ist 24 Stunden im Voraus gültig und bis dahin nicht gültig. Danach läuft sie 8 Stunden ab.

Vorsignierung von POST- und PUT-Anfragen

Viele Operationen, die vorsignierbar sind, benötigen nur eine URL und müssen als HTTP-GET-Anfragen ausgeführt werden. Einige Operationen benötigen jedoch einen Hauptteil und müssen

in einigen Fällen als HTTP-POST-Anforderung oder HTTP-PUT-Anfrage zusammen mit Headern ausgeführt werden. Das Vorsignieren dieser Anfragen ist identisch mit dem Vorsignieren von GET-Anfragen, das Aufrufen der vorab signierten Anforderung ist jedoch komplizierter.

Hier ist ein Beispiel für die Vorsignierung einer S3-Anfrage: `PutObject`

```
val s3 = S3Client.fromEnvironment()

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

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

Die zurückgegebene Datei `HttpRequest` hat einen Methodenwert von `HttpMethod.PUT` und enthält eine URL und Header, die beim future Aufruf der HTTP-Anfrage enthalten sein müssen. Sie können diese Anforderung an einen Aufrufer weiterleiten, der sie in einer anderen Codebasis- oder Programmiersprachenumgebung ausführen kann.

Nachdem Sie die vorsignierte POST- oder PUT-Anforderung erstellt haben, verwenden Sie einen HTTP-Client, um eine Anfrage aufzurufen. Die API zum Aufrufen einer POST- oder PUT-Anforderungs-URL hängt vom verwendeten HTTP-Client ab. Das folgende Beispiel verwendet einen OkHttp HTTP-Client und enthält einen Hauptteil, der Folgendes enthält`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()
```

Operationen, die das SDK vorab signieren kann

Das SDK für Kotlin unterstützt derzeit die Vorsignierung der folgenden API-Operationen, die mit der zugehörigen HTTP-Methode aufgerufen werden müssen.

AWS-Service	Operation	SDK-Erweiterungsmethode	Verwenden Sie die HTTP-Methode
Amazon S3	GetObject	presignGetObject	HTTP GET
Amazon S3	PutObject	presignPutObject	HTTP EINGEBEN
Amazon S3	UploadPart	presignUploadPart	HTTP SETZEN
AWS -Security-Token-Service	GetCallerIdentity	presignGetCallerIdentität	HTTP-BEITRAG
Amazon Polly	SynthesizeSpeech	presignSynthesizeSpeech	HTTP-BEITRAG

Problembehebung FAQs

AWS SDK für Kotlin Bei der Verwendung von in Ihren Anwendungen können einige der in diesem Thema aufgeführten Probleme auftreten. Verwenden Sie die folgenden Vorschläge, um die Ursache zu ermitteln und den Fehler zu beheben.

Wie behebe ich Probleme mit dem Hinweis „Verbindung geschlossen“?

Möglicherweise treten in Ausnahmefällen Probleme mit dem Hinweis „Verbindung geschlossen“ auf, z. B. eines der folgenden Typen:

- `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; HttpErrorCode(CONNECTION_CLOSED)`

Diese Ausnahmen weisen darauf hin, dass eine TCP-Verbindung vom SDK zu einem Dienst unerwartet geschlossen oder zurückgesetzt wurde. Die Verbindung wurde möglicherweise von Ihrem Host, dem AWS Dienst oder einer zwischengeschalteten Partei wie einem NAT-Gateway, einem Proxy oder einem Load Balancer geschlossen.

Diese Arten von Ausnahmen werden automatisch wiederholt, können aber je nach Ihrer Protokollierungskonfiguration immer noch in SDK-Protokollen erscheinen. Wenn die Ausnahme in Ihrem Code ausgelöst wird, bedeutet dies, dass die aktive Wiederholungsstrategie ihre konfigurierten Grenzwerte wie die maximale Anzahl der Versuche oder den Token-Bucket für Wiederholungen ausgeschöpft hat. Weitere Informationen zu Wiederholungsstrategien finden Sie im the section called “Erneute Versuche” Abschnitt dieses Handbuchs. Siehe auch the section called “Warum werden Ausnahmen ausgelöst, bevor die maximale Anzahl an Versuchen erreicht ist?”.

Überwachung von Verbindungen im Leerlauf mit dem OkHttpEngine

Wenn Sie die verwenden `OkHttpEngine` und häufig auf `IOException: unexpected end of stream on <URL>` Ausnahmen stoßen, sollten Sie die Aktivierung der Überwachung inaktiver Verbindungen in Betracht ziehen. Diese Funktion erkennt, wenn Remoteserver Verbindungen geschlossen haben, die sich noch im Verbindungspool befinden, wodurch das Auftreten dieser Ausnahmen reduziert werden kann.

Warum werden Ausnahmen ausgelöst, bevor die maximale Anzahl an Versuchen erreicht ist?

Manchmal werden Ausnahmen angezeigt, von denen Sie erwartet hatten, dass sie wiederholt werden, aber stattdessen ausgelöst wurden. In diesen Situationen können die folgenden Schritte helfen, das Problem zu lösen.

- Stellen Sie sicher, dass die Ausnahme erneut versucht werden kann. Einige Ausnahmen können nicht erneut versucht werden, z. B. solche, die auf eine fehlerhafte Serviceanfrage, fehlende Berechtigungen und nicht vorhandene Ressourcen hinweisen. Das SDK wiederholt diese Art von Ausnahmen nicht automatisch. Informationen darüber, wie Sie nach Ausnahmen suchen, die wiederholt werden können, finden Sie unter. the section called “Prüfen Sie, ob eine Ausnahme erneut versucht werden kann”
- Vergewissern Sie sich, dass die Ausnahme in Ihren Code aufgenommen wird. Einige Ausnahmen erscheinen in Protokollnachrichten als Information, werden aber nicht wirklich in Ihren Code aufgenommen. Beispielsweise können Ausnahmen, die wiederholt werden können, wie Drosselungsfehler, protokolliert werden, da das SDK automatisch mehrere Zyklen durchläuft.

backoff-and-retry Der Aufruf eines SDK-Vorgangs löst nur dann eine Ausnahme aus, wenn er nicht durch die konfigurierten Wiederholungseinstellungen behandelt wurde.

- Überprüfen Sie Ihre konfigurierten Einstellungen für den erneuten Versuch. Weitere Informationen zu Wiederholungsstrategien und Wiederholungsrichtlinien finden Sie im the section called “Erneute Versuche” Abschnitt dieses Handbuchs. Stellen Sie sicher, dass Ihr Code die von Ihnen erwarteten Einstellungen oder die automatischen Standardeinstellungen verwendet.
- Erwägen Sie, Ihre Einstellungen für den erneuten Versuch anzupassen. Nachdem Sie die vorherigen Punkte überprüft haben, das Problem jedoch nicht behoben wurde, sollten Sie erwägen, die Einstellungen für den erneuten Versuch anzupassen.
  - Erhöhen Sie die maximale Anzahl von Versuchen. Standardmäßig beträgt die maximale Anzahl von Versuchen für einen Vorgang 3. Wenn Sie der Meinung sind, dass dies nicht ausreicht und bei der Standardeinstellung immer noch Ausnahmen auftreten, sollten Sie erwägen, die `retryStrategy.maxAttempts` Eigenschaft in Ihrer Client-Konfiguration zu erhöhen. Weitere Informationen finden Sie unter the section called “Maximale Anzahl an Versuchen”.
  - Erhöhen Sie die Verzögerungseinstellungen. Einige Ausnahmen werden möglicherweise zu schnell wiederholt, bevor die Grunderkrankung behoben werden konnte. Wenn Sie vermuten, dass dies der Fall ist, sollten Sie erwägen, die `retryStrategy.delayProvider.maxBackoff` Eigenschaften `retryStrategy.delayProvider.initialDelay` oder in Ihrer Client-Konfiguration zu erhöhen. Weitere Informationen finden Sie unter the section called “Verzögerungen und Backoff”.
  - Deaktivieren Sie den Schutzschaltermodus. Das SDK verwaltet standardmäßig einen Token-Bucket für jeden Service-Client. Wenn das SDK versucht, eine Anfrage zu stellen und diese mit einer wiederholbaren Ausnahme fehlschlägt, wird die Token-Anzahl verringert. Wenn die Anfrage erfolgreich ist, wird die Token-Anzahl erhöht.

    Wenn dieser Token-Bucket 0 verbleibende Token erreicht, ist die Verbindung standardmäßig unterbrochen. Nachdem die Verbindung unterbrochen wurde, deaktiviert das SDK Wiederholungsversuche, und alle aktuellen und nachfolgenden Anfragen, die beim ersten Versuch fehlschlagen, lösen sofort eine Ausnahme aus. Das SDK aktiviert Wiederholungsversuche erneut, nachdem erfolgreiche erste Versuche dem Token-Bucket genügend Kapazität zurückgegeben haben. Dieses Verhalten ist beabsichtigt und wurde entwickelt, um Wiederholungsversuche bei Serviceausfällen und bei Servicewiederherstellungen zu verhindern.

    Wenn Sie es vorziehen, dass das SDK die Wiederholungsversuche bis zur maximal konfigurierten Anzahl von Versuchen fortsetzt, sollten Sie den Circuit Breaker-Modus

deaktivieren, indem Sie die `retryStrategy.tokenBucket.useCircuitBreakerMode` Eigenschaft in Ihrer Client-Konfiguration auf False setzen. Wenn diese Eigenschaft auf false gesetzt ist, wartet der SDK-Client, bis der Token-Bucket genügend Kapazität erreicht hat, anstatt weitere Versuche abzubrechen, die zu einer Ausnahme führen könnten, wenn 0 Token übrig sind.

Wie behebe **`NoSuchMethodError`** ich oder? NoClassDefFoundError

Diese Fehler werden am häufigsten durch fehlende oder widersprüchliche Abhängigkeiten verursacht. Weitere Informationen finden Sie unter the section called “Wie löse ich Abhängigkeitskonflikte?”.

Ich sehe ein für **`NoClassDefFoundError okhttp3/coroutines/ExecuteAsyncKt`**

Dies deutet OkHttp speziell auf ein Abhängigkeitsproblem hin. Weitere Informationen finden Sie unter the section called “Lösung von OkHttp Versionskonflikten in Ihrer Anwendung”.

Wie löse ich Abhängigkeitskonflikte?

Wenn Sie das verwenden AWS SDK für Kotlin, benötigt es bestimmte Abhängigkeiten AWS und Abhängigkeiten von Drittanbietern, damit es ordnungsgemäß funktioniert. Wenn diese Abhängigkeiten fehlen oder zur Laufzeit unerwartete Versionen vorliegen, werden möglicherweise Fehler wie `NoSuchMethodError` oder angezeigt`NoClassDefFoundError`. Diese Abhängigkeitsprobleme lassen sich normalerweise in zwei Gruppen einteilen:

- SDK/Smithy-Abhängigkeitskonflikte
- Konflikte mit Abhängigkeiten von Drittanbietern

Wenn Sie Ihre Kotlin-Anwendung erstellen, werden Sie wahrscheinlich Gradle verwenden, um Abhängigkeiten zu verwalten. Das Hinzufügen einer Abhängigkeit von einem SDK-Serviceclient zu Ihrem Projekt beinhaltet automatisch alle erforderlichen zugehörigen Abhängigkeiten. Wenn Ihre Anwendung jedoch andere Abhängigkeiten hat, können diese mit den vom SDK geforderten Abhängigkeiten kollidieren. Das SDK stützt sich beispielsweise auf OkHttp einen beliebten HTTP-Client, den Ihre Anwendung möglicherweise auch verwendet. Um Ihnen zu helfen, diese Konflikte zu erkennen, bietet Gradle eine praktische Aufgabe, die die Abhängigkeiten Ihres Projekts auflistet:

```
./gradlew dependencies
```

Wenn Sie auf Abhängigkeitskonflikte stoßen, müssen Sie möglicherweise Maßnahmen ergreifen. Sie können entweder eine bestimmte Version einer Abhängigkeit angeben oder Abhängigkeiten in einen lokalen Namespace übertragen. Die Auflösung von Gradle-Abhängigkeiten ist ein komplexes Thema, das in den folgenden Abschnitten des Gradle-Benutzerhandbuchs behandelt wird:

- Die Auflösung von Abhängigkeiten verstehen
- Abhängigkeitsbeschränkungen und Konfliktlösung
- Abstimmung der Abhängigkeitsversionen

Verwaltung von SDK- und Smithy-Abhängigkeiten in Ihrem Projekt

Wenn Sie das SDK verwenden, denken Sie daran, dass seine Module in der Regel von anderen SDK-Modulen mit übereinstimmenden Versionsnummern abhängen. `aws.sdk.kotlin:s3:1.2.3`Hängt beispielsweise von a ab`aws.sdk.kotlin:aws-http:1.2.3`, das von abhängt`aws.sdk.kotlin:aws-core:1.2.3`, und so weiter.

Die SDK-Module verwenden auch spezifische Smithy-Modulversionen. Die Versionen des Smithy-Moduls werden zwar nicht mit den SDK-Versionsnummern synchronisiert, müssen aber mit der erwarteten Version des SDK übereinstimmen. `aws.sdk.kotlin:s3:1.2.3`Könnte zum Beispiel davon abhängen`aws.smithy.kotlin:serde:1.1.1`, was davon abhängt`aws.smithy.kotlin:runtime-core:1.1.1`, und so weiter.

Um Abhängigkeitskonflikte zu vermeiden, aktualisieren Sie alle Ihre SDK-Abhängigkeiten zusammen und tun Sie dasselbe für alle expliziten Smithy-Abhängigkeiten. Erwägen Sie die Verwendung unseres Gradle-Versionskatalogs, um Versionen synchron zu halten und Rätselraten bei der Zuordnung zwischen SDK- und Smithy-Versionen zu vermeiden.

Denken Sie daran, dass kleinere Versionsupdates in SDK/Smithy Modulen grundlegende Änderungen beinhalten können, wie in unseren Versionierungsrichtlinien beschrieben. Lesen Sie beim Upgrade zwischen Nebenversionen sorgfältig die Changelogs und testen Sie das Laufzeitverhalten gründlich.

Lösung von OkHttp Versionskonflikten in Ihrer Anwendung

OkHttpist eine beliebte HTTP-Engine, die das SDK standardmäßig auf JVM verwendet. Ihre Anwendung kann andere Abhängigkeiten oder Frameworks enthalten, die eine andere OkHttp

Version beinhalten. Dies kann dazu führen, dass Klassen im `okhttp3` Namespace nicht richtig angezeigt werden, z. B. `okhttp/coroutines/ExecuteAsyncKt` oder`okhttp3/ConnectionListener`. `NoClassDefFoundError` In diesem Fall sollten Sie generell die neuere Version wählen, um Konflikte zu lösen. Um Ihnen zu helfen, die Ursachen dieser Konflikte aufzuspüren, bietet Gradle eine nützliche Aufgabe. Sie können alle Abhängigkeiten auflisten, indem Sie Folgendes ausführen:

```
./gradlew dependencies
```

Wenn das SDK beispielsweise von einer anderen Abhängigkeit wie Spring Boot abhängt, OkHttp `4.12.0` sollten Sie die verwenden`5.0.0-alpha.14 version`. OkHttp `5.0.0-alpha.14` Sie können dies mit einem `constraints` Block in Gradle tun:

```
dependencies {
    constraints {
        implementation("com.squareup.okhttp3:okhttp:4.12.0")
    }
}
```

Wenn Sie OkHttp 4.x verwenden müssen, bietet das SDK alternativ eine. `OkHttp4Engine` In der Dokumentation finden Sie Informationen zur Konfiguration und Verwendung von Gradle `OkHttp4Engine` in Ihrem Code.

Spott in der AWS SDK für Kotlin

Entwickler können mehrere Frameworks verwenden, um Mocking in Tests mit dem durchzuführen. AWS SDK für Kotlin In diesem Thema werden zusätzliche Konfigurationen oder spezielle Überlegungen beschrieben, die für einige Frameworks erforderlich sind.

MockK

Wenn Sie modulweite Erweiterungsfunktionen mit MockK simulieren, müssen Sie eine zusätzliche Konfiguration vornehmen. Im SDK für Kotlin sind Paginatoren, Waiters und Presigners Beispiele für Erweiterungsfunktionen. Wenn Sie sich also über ihr Verhalten lustig machen wollen, benötigen Sie eine zusätzliche Konfiguration.

Sie müssen anrufen, bevor Sie Ihre Mocks einrichten. `mockkStatic("<MODULE_CLASS_NAME>")` In der Regel lauten die Namen der Modulklassen:

- Paginatoren: `aws.sdk.kotlin.services.<service>.paginators.PaginatorsKt`
- Kellner: `aws.sdk.kotlin.services.<service>.waiters.WaitersKt`
- Presignatoren: `aws.sdk.kotlin.services.<service>.presigners.PresignersKt`

Im folgenden Test, der Mocking `listBucketsPaginated` — eine Funktion zur Erweiterung eines Paginators — beinhaltet, fügen wir beispielsweise hinzu: `mockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")`

```
    @Test
    fun testPaginatedListBuckets() = runTest {

        mockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")
        val s3Client: S3Client = mockk()
        val s3BucketLister = S3BucketLister(s3Client)

        val expectedBuckets = listOf(
            Bucket { name = "bucket1" },
            Bucket { name = "bucket2" }
        )

        val response = ListBucketsResponse { buckets = expectedBuckets }
        coEvery { s3Client.listBucketsPaginated() } returns flowOf(response)

        val result = s3BucketLister.getAllBucketNames()

        assertEquals(listOf("bucket1", "bucket2"), result)
    }
```

Ohne `mockkStatic` wird der folgende Fehler angezeigt:

```
Missing mocked calls inside every { ... } block: make sure the object inside the block
 is a mock
io.mockk.MockKException: Missing mocked calls inside every { ... } block: make sure the
 object inside the block is a mock
    at
 io.mockk.impl.recording.states.StubbingState.checkMissingCalls(StubbingState.kt:14)
    at io.mockk.impl.recording.states.StubbingState.recordingDone(StubbingState.kt:8)
    at io.mockk.impl.recording.CommonCallRecorder.done(CommonCallRecorder.kt:47)
    at io.mockk.impl.eval.RecordedBlockEvaluator.record(RecordedBlockEvaluator.kt:63)
    at io.mockk.impl.eval.EveryBlockEvaluator.every(EveryBlockEvaluator.kt:30)
    at io.mockk.MockKDsl.internalCoEvery(API.kt:100)
```

```
    at io.mockk.MockKKt.coEvery(MockK.kt:174)
```

Bei einer Presigner-Erweiterungsfunktion ohne wird möglicherweise `mockkStatic` Folgendes angezeigt:

```
key is bound to the URI and must not be null
java.lang.IllegalArgumentException: key is bound to the URI and must not be null
    at
 aws.sdk.kotlin.services.s3.serde.GetObjectOperationSerializer.serialize(GetObjectOperationSeri
    at
 aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject(Presigners.kt:49)
    at aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject
$default(Presigners.kt:40)
    at aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject-
exY8QGI(Presigners.kt:30)
```

Alle Beispielartefakte

Code wird getestet

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.paginators.listBucketsPaginated
import kotlinx.coroutines.flow.filter
import kotlinx.coroutines.flow.toList
import kotlinx.coroutines.flow.transform
import kotlinx.coroutines.runBlocking
import org.slf4j.Logger
import org.slf4j.LoggerFactory

fun main() {
    val logger: Logger = LoggerFactory.getLogger(::main.javaClass)

    // Create an S3Client
    S3Client { region = "us-east-1" }.use { s3Client ->
        // Create service instance
        val bucketLister = S3BucketLister(s3Client)
        // Since getAllBucketNames is a suspend function, you'll need to run it in a
 coroutine scope
        runBlocking {
            val bucketNames = bucketLister.getAllBucketNames()
            logger.info("Found buckets: $bucketNames")
        }
    }
```

```
}

class S3BucketLister(private val s3Client: S3Client) {
    suspend fun getAllBucketNames(): List<String> {
        return s3Client.listBucketsPaginated()
            .transform { response ->
                response.buckets?.forEach { bucket ->
                    emit(bucket.name ?: "")
                }
            }
            .filter { it.isNotEmpty() }
            .toList()
    }
}
```

Klasse testen

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.model.Bucket
import aws.sdk.kotlin.services.s3.model.ListBucketsResponse
import aws.sdk.kotlin.services.s3.paginators.listBucketsPaginated
import io.mockk.coEvery
import io.mockk.mockk
import io.mockk.mockkStatic
import kotlinx.coroutines.flow.flowOf
import kotlinx.coroutines.test.runTest
import org.junit.jupiter.api.Assertions.assertEquals
import org.junit.jupiter.api.Test

class S3BucketListerTest {

    @Test
    fun testPaginatedListBuckets() = runTest {

        mockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")
        val s3Client: S3Client = mockk()
        val s3BucketLister = S3BucketLister(s3Client)

        val expectedBuckets = listOf(
            Bucket { name = "bucket1" },
            Bucket { name = "bucket2" }
        )
```

```
        val response = ListBucketsResponse { buckets = expectedBuckets }
        coEvery { s3Client.listBucketsPaginated() } returns flowOf(response)

        val result = s3BucketLister.getAllBucketNames()

        assertEquals(listOf("bucket1", "bucket2"), result)
    }
}
```

build.gradle.kts

```
plugins {
    kotlin("jvm") version "2.1.20"
    application
}

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

repositories {
    mavenCentral()
}

dependencies {
    implementation(platform(awssdk.bom))
    implementation(platform("org.apache.logging.log4j:log4j-bom:2.24.3"))

    implementation(awssdk.services.s3)
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")

    // Testing Dependencies
    testImplementation(platform("org.junit:junit-bom:5.11.0"))
    testImplementation("org.junit.jupiter:junit-jupiter")
    testImplementation("org.jetbrains.kotlinx:kotlinx-coroutines-test:1.7.3")
    testImplementation("io.mockk:mockk:1.14.0")
}

tasks.test {
    useJUnitPlatform()
}

java {
    toolchain {
```

```
        languageVersion = JavaLanguageVersion.of(21)
    }
}

application {
    mainClass = "org.example.S3BucketService"
}
```

Einstellungen.gradle.kts

```
plugins {
    id("org.gradle.toolchains.foojay-resolver-convention") version "0.10.0"
}
rootProject.name = "mockK-static"

dependencyResolutionManagement {
    repositories {
        mavenCentral()
    }

    versionCatalogs {
        create("awssdk") {
            from("aws.sdk.kotlin:version-catalog:1.4.69")
        }
    }
}
```

Arbeiten AWS-Services Sie mit der AWS SDK für Kotlin

Dieses Kapitel enthält Informationen zur Arbeit mit AWS-Services dem SDK für Kotlin.

Inhalt

- Arbeiten Sie mit Amazon S3 mithilfe der AWS SDK für Kotlin
  - Schutz der Datenintegrität mit Prüfsummen
    - Hochladen eines Objekts
      - Verwenden Sie einen vorberechneten Prüfsummenwert
      - Mehrteilige Uploads
    - Herunterladen eines Objekts
      - Asynchrone Validierung
  - Arbeiten Sie mit Amazon S3 Multiregion Access Points mithilfe des SDK für Kotlin
    - Arbeiten Sie mit Access Points für mehrere Regionen
    - Arbeiten Sie mit Objekten und Access Points für mehrere Regionen
- Arbeiten Sie mit DynamoDB unter Verwendung der AWS SDK für Kotlin
  - Verwenden Sie kontobasierte Endpunkte AWS
  - Zuordnen von Klassen zu DynamoDB-Elementen mithilfe des DynamoDB-Mapper (Developer Preview)
    - Erste Schritte mit DynamoDB Mapper
      - Fügen Sie Abhängigkeiten hinzu
      - Erstellen und verwenden Sie einen Mapper
      - Definieren Sie ein Schema mit Klassenanmerkungen
      - Operationen aufrufen
        - Arbeiten Sie mit paginierten Antworten
    - DynamoDB Mapper konfigurieren
      - Verwenden Sie Interzeptoren
        - Verstehen Sie die Anforderungspipeline
        - Hook
          - Nur-Lese-Hooks
          - Hooks ändern

        - Reihenfolge der Ausführung
        - Beispielkonfiguration
- Generieren Sie ein Schema aus Anmerkungen
    - Wenden Sie das Plugin an
    - Konfigurieren des -Plugins
    - Klassen kommentieren
        - Anmerkungen zur Klasse
        - Anmerkungen zu Eigenschaften
    - Definieren Sie einen benutzerdefinierten Artikelkonverter
- Schemas manuell definieren
    - Definieren Sie ein Schema im Code
- Verwenden Sie Sekundärindizes mit DynamoDB Mapper
    - Definieren Sie ein Schema für einen sekundären Index
    - Verwenden Sie Sekundärindizes in Operationen
- Verwenden Sie Ausdrücke
    - Verwenden Sie Ausdrücke in Operationen
    - DSL-Komponenten
        - Attribute
        - Gleichheiten und Ungleichheiten
        - Bereiche und Sätze
        - Boolesche Logik
        - Funktionen und Eigenschaften
        - Schlüsselfilter sortieren

Arbeiten Sie mit Amazon S3 mithilfe der AWS SDK für Kotlin

Ihre Hauptschnittstelle zum Amazon Simple Storage Service für das Kotlin SDK ist der S3Client. Verwenden Sie `S3Client` die gleichen Service-Clients im SDK, um Anfragen an Amazon S3 zu stellen.

Ressourcen, die Ihnen bei der Verwendung des Kotlin-SDK mit S3 helfen, sind:

- die Kotlin SDK API-Referenz für S3.
- das S3-Dienst-Benutzerhandbuch und die Service-API-Referenz.

Die folgenden Themen enthalten geführte Codebeispiele für ausgewählte Kotlin-SDKs APIs , die mit S3 funktionieren.

Themen

- Schutz der Datenintegrität mit Prüfsummen
- Arbeiten Sie mit Amazon S3 Multiregion Access Points mithilfe des SDK für Kotlin

Schutz der Datenintegrität mit Prüfsummen

Amazon Simple Storage Service (Amazon S3) bietet die Möglichkeit, beim Hochladen eines Objekts eine Prüfsumme anzugeben. Wenn Sie eine Prüfsumme angeben, wird diese zusammen mit dem Objekt gespeichert und kann beim Herunterladen des Objekts überprüft werden.

Prüfsummen bieten eine zusätzliche Ebene der Datenintegrität bei der Übertragung von Dateien. Mit Prüfsummen können Sie die Datenkonsistenz überprüfen, indem Sie sicherstellen, dass die empfangene Datei mit der Originaldatei übereinstimmt. Weitere Informationen zu Prüfsummen mit Amazon S3 finden Sie im Amazon Simple Storage Service-Benutzerhandbuch, einschließlich der unterstützten Algorithmen.

Sie haben die Flexibilität, den Algorithmus auszuwählen, der Ihren Anforderungen am besten entspricht, und das SDK die Prüfsumme berechnen zu lassen. Alternativ können Sie mithilfe eines der unterstützten Algorithmen einen vorab berechneten Prüfsummenwert angeben.

> **Note**
>
> Ab Version 1.4.0 von bietet das SDK standardmäßige Integritätsschutzmaßnahmen AWS SDK für Kotlin, indem es automatisch eine Prüfsumme für Uploads berechnet. `CRC32` Das SDK berechnet diese Prüfsumme, wenn Sie keinen vorab berechneten Prüfsummenwert angeben oder wenn Sie keinen Algorithmus angeben, den das SDK zur Berechnung einer Prüfsumme verwenden soll.
> Das SDK bietet auch globale Einstellungen für den Schutz der Datenintegrität, die Sie extern festlegen können. Weitere Informationen finden Sie im Referenzhandbuch und im Tools-Referenzhandbuch.AWS SDKs

Wir behandeln Prüfsummen in zwei Anforderungsphasen: beim Hochladen eines Objekts und beim Herunterladen eines Objekts.

Hochladen eines Objekts

Sie laden Objekte mit dem SDK für Kotlin auf Amazon S3 hoch, indem Sie die `putObject`Funktion mit einem Anforderungsparameter verwenden. Der Anforderungsdatentyp bietet die `checksumAlgorithm` Eigenschaft, die Prüfsummenberechnung zu ermöglichen.

Der folgende Codeausschnitt zeigt eine Anfrage zum Hochladen eines Objekts mit einer Prüfsumme. `CRC32` Wenn das SDK die Anfrage sendet, berechnet es die `CRC32` Prüfsumme und lädt das Objekt hoch. Amazon S3 speichert die Prüfsumme mit dem Objekt.

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

Wenn Sie mit der Anfrage keinen Prüfsummenalgorithmus angeben, variiert das Prüfsummenverhalten je nach der Version des verwendeten SDK, wie in der folgenden Tabelle dargestellt.

Prüfsummenverhalten, wenn kein Prüfsummenalgorithmus bereitgestellt wird

Kotlin SDK-Version	Verhalten der Prüfsumme
früher als 1.4.0	Das SDK berechnet nicht automatisch eine CRC-basierte Prüfsumme und gibt sie in der Anfrage an.
1.4.0 oder höher	Das SDK verwendet den `CRC32` Algorithm us zur Berechnung der Prüfsumme und stellt sie in der Anfrage bereit. Amazon S3 validiert die Integrität der Übertragung, indem es seine eigene `CRC32` Prüfsumme berechnet und sie mit der vom SDK bereitgestellten Prüfsumme vergleicht. Wenn die Prüfsummen übereinst

Kotlin SDK-Version	Verhalten der Prüfsumme
	immen, wird die Prüfsumme zusammen mit dem Objekt gespeichert.

Verwenden Sie einen vorberechneten Prüfsummenwert

Ein mit der Anfrage bereitgestellter vorberechneter Prüfsummenwert deaktiviert die automatische Berechnung durch das SDK und verwendet stattdessen den angegebenen Wert.

Das folgende Beispiel zeigt eine Anfrage mit einer vorberechneten Prüfsumme. SHA256

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

Wenn Amazon S3 feststellt, dass der Prüfsummenwert für den angegebenen Algorithmus falsch ist, gibt der Service eine Fehlerantwort zurück.

Mehrteilige Uploads

Sie können Prüfsummen auch bei mehrteiligen Uploads verwenden.

Sie müssen den Prüfsummenalgorithmus in der Anfrage und in jeder `CreateMultipartUpload` Anfrage angeben. `UploadPart` Als letzten Schritt müssen Sie die Prüfsumme für jeden Teil in der angeben. `CompleteMultipartUpload` Das folgende Beispiel zeigt, wie ein mehrteiliger Upload mit dem angegebenen Prüfsummenalgorithmus erstellt wird.

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

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

```
val completedParts = partFilesToUpload
    .mapIndexed { i, fileName ->
        val uploadPartResponse = s3.uploadPart {
            bucket = "amzn-s3-demo-bucket"
            key = "key"
            body = ByteStream.fromFile(File(fileName))
            uploadId = multipartUpload.uploadId
            partNumber = i + 1 // Part numbers begin at 1.
            checksumAlgorithm = ChecksumAlgorithm.Sha1
        }

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

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

Herunterladen eines Objekts

Wenn Sie die verwenden, um ein Objekt herunterzuladen, validiert das SDK automatisch die Prüfsumme, . `ChecksumMode enabled` wenn die `checksumMode` Eigenschaft des Builders für auf gesetzt `GetObjectRequest` ist`ChecksumMode.Enabled`.

Die Anfrage im folgenden Codeausschnitt weist das SDK an, die Prüfsumme in der Antwort zu validieren, indem es die Prüfsumme berechnet und die Werte vergleicht.

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

> **Note**
>
> Wenn das Objekt nicht mit einer Prüfsumme hochgeladen wurde, findet keine Überprüfung statt.

Wenn Sie eine SDK-Version von 1.4.0 oder höher verwenden, überprüft das SDK automatisch die Integrität von `getObject` Anfragen, ohne der Anfrage etwas `checksumMode = ChecksumMode.Enabled` hinzuzufügen.

Asynchrone Validierung

Da das SDK für Kotlin Streaming-Antworten verwendet, wenn es ein Objekt von Amazon S3 herunterlädt, wird die Prüfsumme berechnet, während Sie das Objekt verwenden. Daher müssen Sie das Objekt verwenden, damit die Prüfsumme validiert wird.

Das folgende Beispiel zeigt, wie eine Prüfsumme validiert wird, indem die Antwort vollständig verarbeitet wird.

```
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.
}
```

Im Gegensatz dazu verwendet der Code im folgenden Beispiel das Objekt in keiner Weise, sodass die Prüfsumme nicht validiert wird.

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

Wenn die vom SDK berechnete Prüfsumme nicht mit der erwarteten Prüfsumme übereinstimmt, die mit der Antwort gesendet wurde, gibt das SDK eine aus. `ChecksumMismatchException`

Arbeiten Sie mit Amazon S3 Multiregion Access Points mithilfe des SDK für Kotlin

Amazon S3 Multiregion Access Points bieten einen globalen Endpunkt, über den Anwendungen Anfragen von Amazon S3 S3-Buckets bearbeiten können, die sich in mehreren befinden. AWS-Regionen Sie können Multi-Region-Access Points verwenden, um multiregionale Anwendungen mit derselben Architektur zu erstellen, die in einer einzelnen Region verwendet wird, und diese Anwendungen dann überall auf der Welt ausführen.

Das Amazon S3 S3-Benutzerhandbuch enthält weitere Hintergrundinformationen zu Multi-Region-Access Points.

Arbeiten Sie mit Access Points für mehrere Regionen

Um einen Access Point mit mehreren Regionen zu erstellen, geben Sie zunächst einen Bucket in jeder AWS Region an, die Sie Anfragen bearbeiten möchten. Das folgende Snippet erstellt zwei Buckets.

Buckets erstellen

Die folgende Funktion erstellt zwei Buckets für die Arbeit mit dem Multi-Region Access Point. Ein Bucket befindet sich in Region `us-east-1` und der andere in Region. `us-west-1`

Die Erstellung des S3-Clients, der als erstes Argument übergeben wurde, ist im ersten Beispiel unter dargestelltthe section called “Mit Objekten arbeiten”.

```
        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.")
            }
        }
```

Sie verwenden den S3-Control-Client des Kotlin SDK, um Multi-Region-Access Points zu erstellen, zu löschen und Informationen darüber abzurufen.

Fügen Sie eine Abhängigkeit vom S3-Steuerartefakt hinzu, wie im folgenden Codeausschnitt gezeigt. (Sie können zu dem *X.Y.Z* Link navigieren, um die neueste verfügbare Version zu sehen.)

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

Konfigurieren Sie den S3-Steuerungsclient so, dass AWS-Region `us-west-2` er damit funktioniert, wie im folgenden Code gezeigt. Alle S3-Control-Client-Operationen müssen auf die `us-west-2` Region abzielen.

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

```
            return s3Control
        }
```

Verwenden Sie den S3-Kontrollclient, um einen Multi-Region-Access Point zu erstellen, indem Sie die (zuvor erstellten) Bucket-Namen angeben, wie im folgenden Code gezeigt.

```
    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"
}
```

Da es sich bei der Erstellung eines Access Points mit mehreren Regionen um einen asynchronen Vorgang handelt, verwenden Sie das Token, das Sie aus der unmittelbaren Antwort erhalten, um den Status des Erstellungsprozesses zu überprüfen. Nachdem bei der Statusüberprüfung eine Erfolgsmeldung zurückgegeben wurde, können Sie den `GetMultiRegionAccessPoint` Vorgang verwenden, um den Alias des Multi-Region Access Points abzurufen. Der Alias ist die letzte Komponente des ARN, die Sie für Operationen auf Objektebene benötigen.

Verwenden Sie das Token, um den Status zu überprüfen

Verwenden Sie den`DescribeMultiRegionAccessPointOperation`, um den Status des letzten Vorgangs zu überprüfen. Sobald der `requestStatus` Wert „SUCCEED“ lautet, können Sie mit dem Multi-Region Access Point arbeiten.

```
    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)
        }
    }
```

Arbeiten Sie mit Objekten und Access Points für mehrere Regionen

Sie verwenden den S3-Client, um mit Objekten in multiregionalen Access Points zu arbeiten. Viele der Operationen, die Sie für Objekte in Buckets verwenden, können Sie auch auf Access Points mit mehreren Regionen verwenden. Weitere Informationen und eine vollständige Liste der Operationen finden Sie unter Kompatibilität von Access Points in mehreren Regionen mit S3-Vorgängen.

Operationen mit Access Points in mehreren Regionen werden mit dem Signaturalgorithmus Asymmetric Sigv4 (SigV4a) signiert. Um SigV4a zu konfigurieren, fügen Sie Ihrem Projekt zunächst die folgenden Abhängigkeiten hinzu. (Sie können zu dem *X.Y.Z* Link navigieren, um die neueste verfügbare Version zu sehen.)

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

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

Nachdem Sie die Abhängigkeiten hinzugefügt haben, konfigurieren Sie den S3-Client so, dass er den SigV4A-Signaturalgorithmus verwendet, wie im folgenden Code gezeigt.

```
    suspend fun createS3Client(): S3Client {
        // Configure your S3Client to use the Asymmetric SigV4 (SigV4a) signing
algorithm.
        val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
        val s3 = S3Client.fromEnvironment {
            authSchemes = listOf(sigV4aScheme)
        }
        return s3
    }
```

Nachdem Sie den S3-Client konfiguriert haben, funktionieren die Operationen, die S3 für Access Points mit mehreren Regionen unterstützt, genauso. Der einzige Unterschied besteht darin, dass der Bucket-Parameter der ARN des Multi-Region Access Points sein muss. Sie können den ARN von der Amazon S3 S3-Konsole oder programmgesteuert abrufen, wie zuvor in der `createMrap` Funktion, die einen ARN zurückgibt, gezeigt wurde.

Das folgende Codebeispiel zeigt den in einer `GetObject` Operation verwendeten ARN.

```
    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
    }
```

Arbeiten Sie mit DynamoDB unter Verwendung der AWS SDK für Kotlin

Verwenden Sie kontobasierte Endpunkte AWS

DynamoDB bietet AWS kontobasierte Endpunkte, die die Leistung verbessern können, indem sie Ihre AWS Konto-ID verwenden, um die Anforderungsweiterleitung zu optimieren.

Um diese Funktion nutzen zu können, müssen Sie Version 1.3.37 oder höher von verwenden. AWS SDK für Kotlin Sie finden die neueste Version des SDK im zentralen Maven-Repository. Sobald eine unterstützte Version des SDK aktiv ist, verwendet sie automatisch die neuen Endpunkte.

Wenn Sie das kontobasierte Routing deaktivieren möchten, haben Sie vier Möglichkeiten:

- Konfigurieren Sie einen DynamoDB-Dienstclient mit der `AccountIdEndpointMode` Einstellung auf. `DISABLED`
- Legen Sie eine Umgebungsvariable fest.
- Legen Sie eine JVM-Systemeigenschaft fest.
- Aktualisieren Sie die Einstellung für die gemeinsam genutzte AWS Konfigurationsdatei.

Der folgende Ausschnitt ist ein Beispiel dafür, wie Sie das kontobasierte Routing deaktivieren können, indem Sie einen DynamoDB-Dienstclient konfigurieren:

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

Das AWS SDKs Referenzhandbuch zu Tools enthält weitere Informationen zu den letzten drei Konfigurationsoptionen.

Zuordnen von Klassen zu DynamoDB-Elementen mithilfe des DynamoDB-Mapper (Developer Preview)

⚠ DynamoDB Mapper ist eine Developer Preview-Version. Die Funktionen sind noch nicht vollständig und können sich ändern.

DynamoDB Mapper ist eine High-Level-Bibliothek, die Mechanismen zur Zuordnung von Kotlin-Klassen zu DynamoDB-Tabellen und -Indizes bietet, ähnlich dem DynamoDB Enhanced Client oder dem AWS SDK für Java Object Persistence Model. AWS SDK für .NET

Sie definieren Schemas, die Ihr Datenobjekt beschreiben und wie Sie es in DynamoDB-Elemente konvertieren. Nachdem Sie das Schema definiert haben, bietet DynamoDB Mapper eine intuitive Oberfläche, über die Sie Ihre Objekte bei Erstellungs-, Lese-, Aktualisierungs- oder Löschvorgängen (CRUD) für Ihre Tabellen und Indizes verwenden können.

Themen

- Erste Schritte mit DynamoDB Mapper
- DynamoDB Mapper konfigurieren
- Generieren Sie ein Schema aus Anmerkungen
- Schemas manuell definieren
- Verwenden Sie Sekundärindizes mit DynamoDB Mapper
- Verwenden Sie Ausdrücke

Erste Schritte mit DynamoDB Mapper

⚠ DynamoDB Mapper ist eine Developer Preview-Version. Die Funktionen sind noch nicht vollständig und können sich ändern.

Das folgende Tutorial stellt die grundlegenden Komponenten von DynamoDB Mapper vor und zeigt, wie Sie ihn in Ihrem Code verwenden können.

Fügen Sie Abhängigkeiten hinzu

Um mit der Arbeit mit DynamoDB Mapper in Ihrem Gradle-Projekt zu beginnen, fügen Sie Ihrer Datei ein Plugin und zwei Abhängigkeiten hinzu. `build.gradle.kts`

(Sie können zu dem *`X.Y.Z`* Link navigieren, um die neueste verfügbare Version zu sehen.)

```
// 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>`*Durch die neueste Version des SDK ersetzen. Die neueste Version des SDK finden Sie unter GitHub.

> **Note**
>
> Einige dieser Abhängigkeiten sind optional, wenn Sie Schemas manuell definieren möchten. the section called “Definieren Sie Schemas manuell”Weitere Informationen und die reduzierten Abhängigkeiten finden Sie unter.

Erstellen und verwenden Sie einen Mapper

DynamoDB Mapper verwendet den DynamoDB-Client AWS SDK für Kotlin von für die Interaktion mit DynamoDB. Sie müssen eine vollständig konfigurierte `DynamoDbClient`Instanz bereitstellen, wenn Sie eine Mapper-Instanz erstellen, wie im folgenden Codeausschnitt gezeigt:

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

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

> **Note**
>
> `DynamoDbMapper`unterstützt keine Operationen zur Tabellenerstellung. Verwenden Sie die`DynamoDbClient`, um Tabellen zu erstellen.

Nachdem Sie die Mapper-Instanz erstellt haben, können Sie sie verwenden, um die Tabelleninstanz abzurufen, wie im Folgenden gezeigt:

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

Der vorherige Code ruft einen Verweis auf eine Tabelle in DynamoDB named `cars` mit einem Schema ab, das von definiert ist `CarSchema` (Schemas werden weiter unten besprochen). Nachdem Sie eine Tabelleninstanz erstellt haben, können Sie Operationen an ihr ausführen. Der folgende Codeausschnitt zeigt zwei Beispieloperationen für die `cars` Tabelle:

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

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

Der vorherige Code erstellt ein neues Element in der `cars` Tabelle. Der Code erstellt eine `Car` Inline-Instanz unter Verwendung der `Car` Klasse, deren Definition unten gezeigt wird. Als Nächstes fragt der Code die `cars` Tabelle nach Elementen ab, deren Partitionsschlüssel lautet, `Peugeot` und druckt sie aus. Die Operationen werden im Folgenden ausführlicher beschrieben.

Definieren Sie ein Schema mit Klassenanmerkungen

Für eine Vielzahl von Kotlin-Klassen kann das SDK zur Build-Zeit mithilfe des DynamoDB Mapper Schema Generator-Plug-ins für Gradle automatisch Schemas generieren. Wenn Sie den Schema-Generator verwenden, überprüft das SDK Ihre Klassen, um daraus das Schema abzuleiten, wodurch einige der Standardvorgaben, die mit der manuellen Definition von Schemas verbunden sind, vereinfacht werden. Sie können das generierte Schema mithilfe zusätzlicher Anmerkungen und Konfigurationen anpassen.

Um ein Schema aus Anmerkungen zu generieren, kommentieren Sie zunächst Ihre Klassen mit @`DynamoDbItem` und alle Schlüssel mit und. @`DynamoDbPartitionKey` @`DynamoDbSortKey` Der folgende Code zeigt die `Car` annotierte Klasse:

```
// 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
)
```

Nach dem Erstellen können Sie auf die automatisch generierte `CarSchema` Datei zurückgreifen. Sie können die Referenz in der `getTable` Mapper-Methode verwenden, um eine Tabelleninstanz abzurufen, wie im Folgenden gezeigt:

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

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

Alternativ können Sie die Tabelleninstanz abrufen, indem Sie eine Erweiterungsmethode nutzen`DynamoDbMapper`, die beim Erstellen automatisch generiert wird. Wenn Sie diesen Ansatz verwenden, müssen Sie nicht namentlich auf das Schema verweisen. Wie im Folgenden gezeigt, gibt die automatisch generierte `getCarsTable` Erweiterungsmethode einen Verweis auf die Tabelleninstanz zurück:

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

Weitere Einzelheiten und Beispiele finden Sie unter the section called “Generieren Sie ein Schema”.

Operationen aufrufen

DynamoDB Mapper unterstützt eine Teilmenge der in den SDKs verfügbaren Operationen. `DynamoDbClient` Mapper-Operationen werden genauso benannt wie ihre Gegenstücke auf dem SDK-Client. Viele request/response Mapper-Mitglieder sind dieselben wie ihre SDK-Client-Gegenstücke, obwohl einige umbenannt, neu eingegeben oder ganz gelöscht wurden.

Sie rufen einen Vorgang auf einer Tabelleninstanz mithilfe einer DSL-Syntax auf, wie im Folgenden dargestellt:

```
import aws.sdk.kotlin.hll.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)
```

Sie können einen Vorgang auch aufrufen, indem Sie ein explizites Anforderungsobjekt verwenden:

```
import aws.sdk.kotlin.hll.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)
```

Die beiden vorherigen Codebeispiele sind gleichwertig.

Arbeiten Sie mit paginierten Antworten

Bei einigen `query` Vorgängen `scan` können Datensammlungen zurückgegeben werden, die möglicherweise zu umfangreich sind, um sie in einer einzigen Antwort zurückzugeben. Um sicherzustellen, dass alle Objekte verarbeitet werden, bietet DynamoDB Mapper Paginierungsmethoden, die DynamoDB nicht sofort aufrufen, sondern stattdessen a `Flow` vom Typ Operation Response zurückgeben, wie im Folgenden dargestellt: `Flow<ScanResponse<Car>>`

```
import aws.sdk.kotlin.hll.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) }
}
```

Oft ist ein Objektfluss für die Geschäftslogik nützlicher als ein Antwortfluss, der Objekte enthält. Der Mapper bietet eine Erweiterungsmethode für paginierte Antworten, um auf den Objektfluss zuzugreifen. Der folgende Code gibt beispielsweise a und `Flow<Car>` nicht a zurück, `Flow<ScanResponse<Car>>` wie zuvor gezeigt:

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

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

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

DynamoDB Mapper konfigurieren

⚠ DynamoDB Mapper ist eine Developer Preview-Version. Die Funktionen sind noch nicht vollständig und können sich ändern.

DynamoDB Mapper bietet Konfigurationsoptionen, mit denen Sie das Verhalten der Bibliothek an Ihre Anwendung anpassen können.

Verwenden Sie Interzeptoren

Die DynamoDB Mapper-Bibliothek definiert Hooks, auf die Sie in kritischen Phasen der Mapper-Anforderungspipeline zurückgreifen können. Sie können die `Interceptor` Schnittstelle implementieren, um Hooks zur Beobachtung oder Änderung des Mapper-Prozesses zu implementieren.

Sie können einen oder mehrere Interceptoren auf einem einzelnen DynamoDB-Mapper als Konfigurationsoption registrieren. Im Beispiel am Ende dieses Abschnitts erfahren Sie, wie Sie einen Interceptor registrieren.

Verstehen Sie die Anforderungspipeline

Die Anforderungspipeline des Mappers besteht aus den folgenden 5 Schritten:

1. Initialisierung: Richten Sie den Vorgang ein und erfassen Sie den ersten Kontext.
2. Serialisierung: Konvertiert Anforderungsobjekte auf hoher Ebene in Anforderungsobjekte auf niedriger Ebene. Dieser Schritt konvertiert Kotlin-Objekte auf hoher Ebene in DynamoDB-Elemente, die aus Attributnamen und -werten bestehen.
3. Low-Level-Aufruf: Führt eine Anfrage auf dem zugrunde liegenden DynamoDB-Client aus.
4. Deserialisierung: Konvertiert Antwortobjekte auf niedriger Ebene in Antwortobjekte auf hoher Ebene. Dieser Schritt beinhaltet die Konvertierung von DynamoDB-Elementen, die aus Attributnamen und -werten bestehen, in Kotlin-Objekte auf hoher Ebene.
5. Abschluss: Finalisieren Sie die Antwort auf hoher Ebene, um zum Anrufer zurückzukehren. Wenn während der Ausführung der Pipeline eine Ausnahme ausgelöst wurde, schließt dieser Schritt die Ausnahme ab, die an den Aufrufer ausgelöst wurde.

Hook

Hooks sind Interceptormethoden, die der Mapper vor oder nach bestimmten Schritten in der Pipeline aufruft. Es gibt zwei Varianten von Hooks: Read-Only und Modify (oder Read-Write). Dies `readBeforeInvocation` ist beispielsweise ein schreibgeschützter Hook, den der Mapper in der Phase vor dem Aufrufschritt auf niedriger Ebene ausführt.

Nur-Lese-Hooks

Der Mapper ruft vor und nach jedem Schritt in der Pipeline schreibgeschützte Hooks auf (außer vor dem Initialisierungsschritt und nach dem Abschlussschritt). Schreibgeschützte Hoods bieten eine schreibgeschützte Ansicht eines laufenden Vorgangs auf hoher Ebene. Sie bieten einen Mechanismus, mit dem der Status eines Vorgangs untersucht werden kann, z. B. zum Protokollieren, Debuggen und Sammeln von Metriken. Jeder nur lesbare Hook erhält ein Kontextargument und kehrt zurück. `Unit`

Der Mapper fängt jede Ausnahme ab, die bei einem schreibgeschützten Hook ausgelöst wird, und fügt sie dem Kontext hinzu. Anschließend leitet er den Kontext mit der Ausnahme an nachfolgende Interceptor-Hooks in derselben Phase weiter. Der Mapper gibt dem Aufrufer erst dann eine Ausnahme, wenn er den schreibgeschützten Hook des letzten Interceptors für dieselbe Phase aufgerufen hat. Wenn ein Mapper beispielsweise mit zwei Interceptoren konfiguriert ist `A` und sein Hook eine Ausnahme auslöst`B`, fügt `A` der Mapper die Ausnahme zu dem an den `readAfterSerialization` Hook übergebenen Kontext hinzu. `B` `readAfterSerialization` Nachdem `B` der `readAfterSerialization` Hook abgeschlossen ist, gibt der Mapper die Ausnahme zurück an den Aufrufer.

Hooks ändern

Der Mapper ruft Modif-Hooks vor jedem Schritt in der Pipeline auf (außer vor der Initialisierung). Modify-Hooks bieten die Möglichkeit, einen laufenden Vorgang auf hoher Ebene zu sehen und zu ändern. Sie können verwendet werden, um Verhalten und Daten auf eine Weise anzupassen, die Mapper-Konfiguration und Elementschemas nicht bieten. Jeder Modify-Hook erhält ein Kontextargument und gibt als Ergebnis eine Teilmenge dieses Kontextes zurück — entweder durch den Hook modifiziert oder aus dem Eingabekontext übernommen.

Wenn der Mapper bei der Ausführung eines Modif-Hooks eine Ausnahme abfängt, führt er in derselben Phase keine Modif-Hooks anderer Interceptoren aus. Der Mapper fügt die Ausnahme dem Kontext hinzu und übergibt sie an den nächsten schreibgeschützten Hook. Der Mapper gibt dem Aufrufer erst dann eine Ausnahme, wenn er den schreibgeschützten Hook des letzten Interceptors für

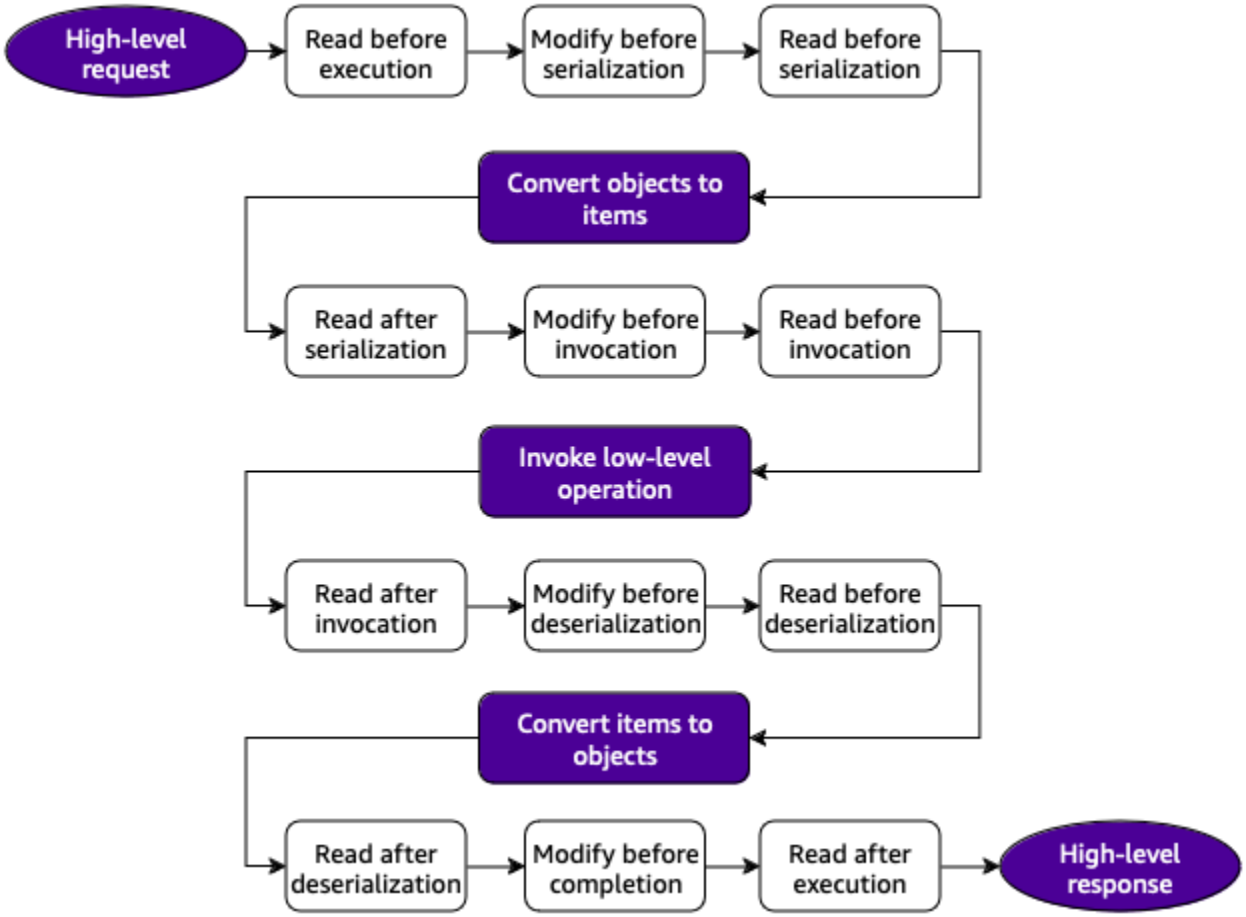
dieselbe Phase aufgerufen hat. Wenn ein Mapper beispielsweise mit zwei Interceptoren konfiguriert ist `A` und sein Hook eine Ausnahme auslöst`B`, wird `A` der `modifyBeforeSerialization` Hook nicht aufgerufen. `B` `modifyBeforeSerialization` Der `readAfterSerialization` Hook von `A` Interceptors und `B'` S werden ausgeführt. Danach wird die Ausnahme an den Aufrufer zurückgegeben.

Reihenfolge der Ausführung

Die Reihenfolge, in der Interceptors in der Konfiguration eines Mappers definiert sind, bestimmt die Reihenfolge, in der der Mapper die Hooks aufruft:

- Für Phasen vor dem Aufrufschritt auf niedriger Ebene werden Hooks in derselben Reihenfolge ausgeführt, in der sie der Konfiguration hinzugefügt wurden.
- Für Phasen nach dem Aufrufschritt auf niedriger Ebene werden Hooks in umgekehrter Reihenfolge ausgeführt als in der Reihenfolge, in der sie der Konfiguration hinzugefügt wurden.

Das folgende Diagramm zeigt die Ausführungsreihenfolge der Hook-Methoden:



Textbeschreibung der Ausführungsreihenfolge der Hook-Methoden

Ein Mapper führt die Hooks eines Interceptors in der folgenden Reihenfolge aus:

1. DynamoDB Mapper ruft eine Anforderung auf hoher Ebene auf
2. Vor der Ausführung lesen
3. Vor der Serialisierung ändern
4. Vor der Serialisierung lesen
5. DynamoDB Mapper konvertiert Objekte in Elemente
6. Nach der Serialisierung lesen
7. Vor dem Aufruf ändern
8. Vor dem Aufruf lesen
9. DynamoDB Mapper ruft die Low-Level-Operation auf
10. Nach dem Aufruf lesen
11. Vor der Deserialisierung ändern
12. Vor der Deserialisierung lesen
13. DynamoDB Mapper konvertiert Elemente in Objekte
14. Nach der Deserialisierung lesen
15. Vor der Fertigstellung ändern
16. Nach der Ausführung lesen
17. DynamoDB Mapper gibt eine Antwort auf hoher Ebene zurück

Beispielkonfiguration

Das folgende Beispiel zeigt, wie ein Interceptor auf einer Instance konfiguriert wird: `DynamoDbMapper`

```
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
}
```

Generieren Sie ein Schema aus Anmerkungen

⚠ DynamoDB Mapper ist eine Developer Preview-Version. Die Funktionen sind noch nicht vollständig und können sich ändern.

DynamoDB Mapper stützt sich auf Schemas, die die Zuordnung zwischen Ihren Kotlin-Klassen und DynamoDB-Elementen definieren. Ihre Kotlin-Klassen können die Erstellung von Schemas mithilfe des Gradle-Plug-ins für den Schemagenator vorantreiben.

Wenden Sie das Plugin an

Um mit der Codegenerierung von Schemas für Ihre Klassen zu beginnen, wenden Sie das Plugin im Build-Skript Ihrer Anwendung an und fügen Sie eine Abhängigkeit vom Annotationsmodul hinzu. Der folgende Gradle-Skriptausschnitt zeigt das notwendige Setup für die Codegenerierung.

(Sie können zum *X.Y.Z* Link navigieren, um die neueste verfügbare Version zu sehen.)

```
// 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")
}
```

Konfigurieren des -Plugins

Das Plugin bietet eine Reihe von Konfigurationsoptionen, die Sie mithilfe der `dynamoDbMapper { ... }` Plugin-Erweiterung in Ihrem Build-Skript anwenden können:

Option	Beschreibung der Option	Werte
`generateBuilderClasses`	Steuert, ob Builder-Klassen im DSL-Stil für Klassen generiert werden, die mit annotiert sind `@DynamoDbItem`	`WHEN_REQUIRED` (Standard): Builder-Klassen werden nicht für Klassen generiert, die nur aus öffentlichen, veränderbaren Mitgliedern bestehen und einen Null-Arg-Konstruktor haben `ALWAYS`: Builder-Klassen werden immer generiert
`visibility`	Steuert die Sichtbarkeit der generierten Klassen	`PUBLIC` (Standard) `INTERNAL`
`destinationPackage`	Gibt den Paketnamen für generierte Klassen an	`RELATIVE`(Standard): Schemaklassen werden in einem Unterpaket relativ zu Ihrer annotierten Klasse generiert. Standardmäßig hat das Unterpaket einen Namen`dynamodbmapper.generatedschemas` , und dies kann durch Übergabe eines Zeichenkettenparameters konfiguriert werden

Option	Beschreibung der Option	Werte
		`ABSOLUTE`: Schemaklassen werden in einem absoluten Paket relativ zum Stamm Ihrer Anwendung generiert. Standardmäßig hat das Paket einen Namen`aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas` , und dies kann durch Übergabe eines Zeichenkettenparameters konfiguriert werden.
`generateGetTableExtension`	Steuert, ob eine `DynamoDbMapper.get${CLASS_NAME}Table` Erweiterungsmethode generiert wird	`true` (Standard) `false`

Example Beispiel für eine Plugin-Konfiguration zur Codegenerierung

Das folgende Beispiel konfiguriert das Zielpaket und die Sichtbarkeit des generierten Schemas:

```
// build.gradle.kts

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

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

Klassen kommentieren

Der Schema-Generator sucht nach Klassenanmerkungen, um zu ermitteln, für welche Klassen Schemas generiert werden sollen. Um sich für die Generierung von Schemas zu entscheiden, kommentieren Sie Ihre Klassen mit. @`DynamoDbItem` Sie müssen auch eine Klasseneigenschaft, die als Partitionsschlüssel des Elements dient, mit der Anmerkung annotieren. @`DynamoDbPartitionKey`

Die folgende Klassendefinition zeigt die minimal erforderlichen Anmerkungen für die Schemagenerierung:

Example

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

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

Anmerkungen zur Klasse

Die folgenden Anmerkungen werden auf Klassen angewendet, um die Schemagenerierung zu steuern:

- `@DynamoDbItem`: Gibt an, dass dies einen Elementtyp in einer Tabelle class/interface beschreibt. Alle öffentlichen Eigenschaften dieses Typs werden Attributen zugeordnet, sofern sie nicht ausdrücklich ignoriert werden. Falls vorhanden, wird ein Schema für diese Klasse generiert.
  - `converterName`: Ein optionaler Parameter, der angibt, dass ein benutzerdefiniertes Schema verwendet werden sollte und nicht das, das vom Schema-Generator-Plugin erstellt wurde. Dies ist der vollständig qualifizierte Name der benutzerdefinierten `ItemConverter` Klasse. Der the section called “Definieren Sie einen benutzerdefinierten Artikelkonverter” Abschnitt zeigt ein Beispiel für die Erstellung und Verwendung eines benutzerdefinierten Schemas.

Anmerkungen zu Eigenschaften

Sie können die folgenden Anmerkungen auf Klasseneigenschaften anwenden, um die Schemagenerierung zu steuern:

- @DynamoDbPartitionKey: Gibt den Partitionsschlüssel für das Element an.
- @DynamoDbSortKey: Gibt einen optionalen Sortierschlüssel für das Element an.
- @DynamoDbIgnore: Gibt an, dass diese Klasseneigenschaft vom DynamoDB-Mapper nicht in to/from ein Item-Attribut umgewandelt werden soll.
- @DynamoDbAttribute: Gibt einen optionalen benutzerdefinierten Attributnamen für diese Klasseneigenschaft an.

Definieren Sie einen benutzerdefinierten Artikelkonverter

In einigen Fällen möchten Sie vielleicht einen benutzerdefinierten Artikelkonverter für Ihre Klasse definieren. Ein Grund dafür wäre, wenn Ihre Klasse einen Typ verwendet, der vom Schema-Generator-Plugin nicht unterstützt wird. Wir verwenden die folgende Version der `Employee` Klasse als Beispiel:

```
import kotlin.uuid.Uuid

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

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

Die `Employee` Klasse verwendet jetzt einen `kotlin.uuid.Uuid` Typ, der derzeit vom Schema-Generator nicht unterstützt wird. Die Schemagenerierung schlägt mit einem Fehler fehl:`Unsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)`. Dieser Fehler weist darauf hin, dass das Plugin keinen Elementkonverter für diese Klasse generieren kann. Deshalb müssen wir unsere eigenen schreiben.

Dazu implementieren wir eine `ItemConverter` für die Klasse und ändern dann die Klassenanmerkung, indem wir den vollqualifizierten Namen des neuen Elementkonverters angeben. `@DynamoDbItem`

Zuerst implementieren wir a `ValueConverter` für die `kotlin.uuid.Uuid` Klasse:

```
import aws.sdk.kotlin.hll.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())
}
```

Dann implementieren wir an `ItemConverter` für unsere `Employee` Klasse. Der `ItemConverter` verwendet diesen neuen Wertekonverter im Attributdeskriptor für „WorkstationID“:

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.AttributeDescriptor
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.SimpleItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.values.scalars.IntConverter
import aws.sdk.kotlin.hll.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
        )
    ),
)
```

Nachdem wir den Elementkonverter definiert haben, können wir ihn auf unsere Klasse anwenden. Wir aktualisieren die @DynamoDbItem Anmerkung so, dass sie auf den Elementkonverter verweist, indem wir den vollqualifizierten Klassennamen angeben, wie im Folgenden dargestellt:

```
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
)
```

Endlich können wir beginnen, die Klasse mit DynamoDB Mapper zu verwenden.

Schemas manuell definieren

⚠ DynamoDB Mapper ist eine Developer Preview-Version. Die Funktionen sind noch nicht vollständig und können sich ändern.

Definieren Sie ein Schema im Code

Für maximale Kontrolle und Anpassungsfähigkeit können Sie Schemas im Code manuell definieren und anpassen.

Wie im folgenden Codeausschnitt gezeigt, müssen Sie weniger Abhängigkeiten in Ihre `build.gradle.kts` Datei aufnehmen als bei der Verwendung der annotierungsgesteuerten Schemaerstellung.

(Sie können zu dem *X.Y.Z* Link navigieren, um die neueste verfügbare Version zu sehen.)

```
// 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.
}
```

Beachten Sie, dass Sie weder das Schema-Generator-Plugin noch das Annotationspaket benötigen.

Die Zuordnung zwischen einer Kotlin-Klasse und einem DynamoDB-Element erfordert eine `ItemSchema<T>` Implementierung, wobei der Typ der Kotlin-Klasse T ist. Ein Schema besteht aus den folgenden Elementen:

- Ein Elementkonverter, der definiert, wie zwischen Kotlin-Objektinstanzen und DynamoDB-Elementen konvertiert wird.
- Eine Partitionsschlüsselspezifikation, die den Namen und den Typ des Partitionsschlüsselattributs definiert.
- Optional eine Sortierschlüsselspezifikation, die den Namen und den Typ des Sortierschlüsselattributs definiert.

Im folgenden Code erstellen wir manuell eine `CarSchema` Instanz:

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.hll.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)
```

Der vorherige Code erstellt einen Konverter mit dem Namen`carConverter`, der als anonyme Implementierung von definiert ist`ItemConverter<Car>`. Die `convertTo` Methode des Konverters akzeptiert ein `Car` Argument und gibt eine `Item` Instanz zurück, die die Literalschlüssel und Werte von DynamoDB-Elementattributen darstellt. Die `convertFrom` Methode des Konverters akzeptiert ein `Item` Argument und gibt anhand der Attributwerte des Arguments eine `Car` Instanz zurück. `Item`

Als Nächstes erstellt der Code zwei Schlüsselspezifikationen: eine für den Partitionsschlüssel und eine für den Sortierschlüssel. Jede DynamoDB-Tabelle oder jeder DynamoDB-Index muss genau einen Partitionsschlüssel haben, und entsprechend muss auch jede DynamoDB-Mapper-Schemadefinition über genau einen Partitionsschlüssel verfügen. Schemas können auch einen Sortierschlüssel haben.

In der letzten Anweisung erstellt der Code anhand des Konverters und der Schlüsselspezifikationen ein Schema für die `cars` DynamoDB-Tabelle.

Das resultierende Schema entspricht dem durch Anmerkungen gesteuerten Schema, das wir in diesem Abschnitt generiert haben. the section called “Definieren Sie ein Schema mit Klassenanmerkungen” Als Referenz ist die folgende Klasse mit Anmerkungen aufgeführt, die wir verwendet haben:

Autoklasse mit DynamoDB-Mapper-Anmerkungen

```
@DynamoDbItem
data class Car(
    @DynamoDbPartitionKey
```

```
    val make: String,

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

Neben der Implementierung Ihrer eigenen `ItemConverter` enthält DynamoDB Mapper mehrere hilfreiche Implementierungen wie:

- `SimpleItemConverter`: bietet eine einfache Konvertierungslogik mithilfe einer Builder-Klasse und Attributdeskriptoren. Sehen Sie sich das Beispiel in der the section called “Definieren Sie einen benutzerdefinierten Artikelkonverter” an, wie Sie diese Implementierung nutzen können.
- `HeterogeneousItemConverter`: stellt eine polymorphe Typkonvertierungslogik bereit, indem es ein Diskriminatorattribut und `ItemConverter` Delegatinstanzen für Subtypen verwendet.
- `DocumentConverter`: bietet Konvertierungslogik für unstrukturierte Daten in Objekten. `Document`

Verwenden Sie Sekundärindizes mit DynamoDB Mapper

> ⚠ DynamoDB Mapper ist eine Developer Preview-Version. Die Funktionen sind noch nicht vollständig und können sich ändern.

Definieren Sie ein Schema für einen sekundären Index

DynamoDB-Tabellen unterstützen Sekundärindizes, die den Zugriff auf Daten mit anderen Schlüsseln als denen ermöglichen, die in der Basistabelle selbst definiert sind. Wie Basistabellen interagiert DynamoDB Mapper mithilfe des Typs mit Indizes. `ItemSchema`

DynamoDB-Sekundärindizes müssen nicht jedes Attribut aus der Basistabelle enthalten. Dementsprechend kann sich die Kotlin-Klasse, die einem Index zugeordnet ist, von der Kotlin-Klasse unterscheiden, die der Basistabelle dieses Indexes zugeordnet ist. Wenn das der Fall ist, muss ein separates Schema für die Indexklasse deklariert werden.

Der folgende Code erstellt manuell ein Indexschema für die DynamoDB-Tabelle`cars`.

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
```

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

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

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

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

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

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

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

            @DynamoDbSortKey
            val model: String,

            val initialYear: Int
        )
*/
```

Wir können jetzt `Model` Instanzen in Operationen verwenden.

Verwenden Sie Sekundärindizes in Operationen

DynamoDB Mapper unterstützt eine Teilmenge von Operationen für Indizes, nämlich und. `queryPaginated` `scanPaginated` Um diese Operationen für einen Index aufzurufen, müssen Sie zunächst einen Verweis auf einen Index aus dem Tabellenobjekt abrufen. Im folgenden Beispiel verwenden wir `modelSchema` das, was wir zuvor für den `cars-by-model` Index erstellt haben (Erstellung hier nicht gezeigt):

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

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

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

Verwenden Sie Ausdrücke

> ⚠ DynamoDB Mapper ist eine Developer Preview-Version. Die Funktionen sind noch nicht vollständig und können sich ändern.

Bestimmte DynamoDB-Operationen akzeptieren Ausdrücke, mit denen Sie Einschränkungen oder Bedingungen angeben können. DynamoDB Mapper bietet eine idiomatische Kotlin-DSL zum Erstellen von Ausdrücken. Die DSL verleiht Ihrem Code mehr Struktur und Lesbarkeit und erleichtert auch das Schreiben von Ausdrücken.

In diesem Abschnitt wird die DSL-Syntax beschrieben und es werden verschiedene Beispiele bereitgestellt.

Verwenden Sie Ausdrücke in Operationen

Sie verwenden Ausdrücke beispielsweise in Operationen`scan`, bei denen die zurückgegebenen Elemente anhand von von Ihnen definierter Kriterien gefiltert werden. Um Ausdrücke mit DynamoDB Mapper zu verwenden, fügen Sie die Ausdruckskomponente zur Operationsanforderung hinzu.

Der folgende Ausschnitt zeigt ein Beispiel für einen Filterausdruck, der in einer Operation verwendet wird. `scan` Es verwendet ein Lambda-Argument, um die Filterkriterien zu beschreiben, die die zurückzugebenden Elemente auf Elemente mit dem `year` Attributwert 2001 beschränken:

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

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

Das folgende Beispiel zeigt eine `query` Operation, die Ausdrücke an zwei Stellen unterstützt: Sortierschlüsselfilterung und Nichtschlüsselfilterung:

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

Der vorherige Code filtert Ergebnisse nach Ergebnissen, die alle drei Kriterien erfüllen:

- Der Wert des Partitionsschlüsselattributs ist 1000 -AND-
- Der Wert des Sortierschlüsselattributs beginnt mit dem Buchstaben M -AND -
- Der Wert des Attributs für das Jahr ist 2001

DSL-Komponenten

Die DSL-Syntax macht verschiedene Komponententypen verfügbar (siehe unten), die Sie zum Erstellen von Ausdrücken verwenden.

Attribute

Die meisten Bedingungen verweisen auf Attribute, die durch ihren Schlüssel oder Dokumentpfad identifiziert werden. Mit dem DSK erstellen Sie alle Attributreferenzen mithilfe der `attr` Funktion und nehmen optional weitere Änderungen vor.

Der folgende Code zeigt eine Reihe von einfachen bis komplexen Beispielattributen, z. B. die Dereferenzierung von Listen nach Index und die Dereferenzierung von Zuordnungen nach Schlüssel:

```
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`.
```

Gleichheiten und Ungleichheiten

Sie können Attributwerte in einem Ausdruck anhand von Gleichheiten und Ungleichheiten vergleichen. Sie können Attributwerte mit Literalwerten oder anderen Attributwerten vergleichen. Die Funktionen, mit denen Sie die Bedingungen angeben, sind:

- `eq`: ist gleich (entspricht==)
- `neq`: ist nicht gleich (entspricht!=)
- `gt`: ist größer als (entspricht>)
- `gte`: ist größer als oder gleich (entspricht>=)
- `lt`: ist kleiner als (entspricht<)
- `lte`: ist kleiner oder gleich (entspricht<=)

Sie kombinieren die Vergleichsfunktion mit Argumenten, indem Sie die Infix-Notation verwenden, wie in den folgenden Beispielen gezeigt:

```
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`.
```

Bereiche und Sätze

Zusätzlich zu Einzelwerten können Sie Attributwerte mit mehreren Werten in Bereichen oder Sätzen vergleichen. Sie verwenden die `isIn` Infix-Funktion, um den Vergleich durchzuführen, wie in den folgenden Beispielen gezeigt:

```
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",
)
```

Die `isIn` Funktion bietet Überladungen für Sammlungen (wie`Set<String>`) und für Grenzen, die Sie als Kotlin ausdrücken können `ClosedRange<T>` (z. B.). `IntRange` Für Grenzen, die Sie nicht als a ausdrücken können `ClosedRange<T>` (wie Byte-Arrays oder andere Attributverweise), können Sie die Funktion verwenden: `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`.
```

Boolesche Logik

Sie können einzelne Bedingungen kombinieren oder mithilfe der booleschen Logik ändern, indem Sie die folgenden Funktionen verwenden:

- `and`: Jede Bedingung muss wahr sein (entspricht `&&`)
- `or`: mindestens eine Bedingung muss wahr sein (entspricht `||`)
- `not`: Die angegebene Bedingung muss falsch sein (entspricht `!`)

Die folgenden Beispiele zeigen jede Funktion:

```
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.
)
```

Sie können boolesche Bedingungen weiter mit booleschen Funktionen kombinieren, um verschachtelte Logik zu erstellen, wie im folgenden Ausdruck dargestellt:

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

Der vorherige Ausdruck filtert Ergebnisse nach Ergebnissen, die eine der folgenden Bedingungen erfüllen:

- Beide Bedingungen sind wahr:
  - `foo`Der Attributwert ist 123 -AND-
  - `bar`Der Attributwert ist „abc“
- Beide Bedingungen sind wahr:
  - `foo`Der Attributwert ist 234 -AND-
  - `bar`Der Attributwert ist „bcd“

Dies entspricht dem folgenden booleschen Kotlin-Ausdruck:

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

Funktionen und Eigenschaften

Die folgenden Funktionen und Eigenschaften bieten zusätzliche Ausdrucksmöglichkeiten:

- `contains`: prüft, ob ein string/list Attributwert einen bestimmten Wert enthält
- `exists`: prüft, ob ein Attribut definiert ist und einen beliebigen Wert enthält (einschließlich`null`)
- `notExists`: prüft, ob ein Attribut undefiniert ist
- `isOfType`: prüft, ob ein Attributwert einen bestimmten Typ hat, z. B. eine Zeichenfolge, eine Zahl, einen booleschen Wert usw.
- `size`: ermittelt die Größe eines Attributs, z. B. die Anzahl der Elemente in einer Sammlung oder die Länge einer Zeichenfolge
- `startsWith`: prüft, ob der Wert eines String-Attributs mit einer bestimmten Teilzeichenfolge beginnt

Die folgenden Beispiele zeigen die Verwendung zusätzlicher Funktionen und Eigenschaften, die Sie in Ausdrücken verwenden können:

```
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.
```

Schlüsselfilter sortieren

Filterausdrücke für Sortierschlüssel (z. B. im `keyCondition` Parameter der `query` Operation) verwenden keine benannten Attributwerte. Um einen Sortierschlüssel in einem Filter zu verwenden, müssen Sie das Schlüsselwort `sortKey` in allen Vergleichen verwenden. Das `sortKey` Schlüsselwort ersetzt`attr("<sort key name>")`, wie in den folgenden Beispielen gezeigt:

```
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).
```

Sie können Sortierschlüsselfilter nicht mit boolescher Logik kombinieren und sie unterstützen nur eine Teilmenge der oben beschriebenen Vergleiche:

- Gleichheiten und Ungleichheiten: Alle Vergleiche werden unterstützt
- Bereiche und Gruppen: Alle Vergleiche werden unterstützt
- Boolesche Logik: wird nicht unterstützt
- Funktionen und Eigenschaften: wird nur unterstützt `startsWith`

SDK für Kotlin-Codebeispiele

Die Codebeispiele in diesem Thema zeigen Ihnen, wie Sie das AWS SDK für Kotlin mit verwenden. AWS

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder dienstspezifische Funktionen genutzt werden können.

Dienstleistungen

- API-Gateway-Beispiele unter Verwendung von SDK für Kotlin
- Aurora-Beispiele unter Verwendung von SDK für Kotlin
- Auto-Scaling-Beispiele unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Bedrock unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Kotlin
- CloudWatch Beispiele für die Verwendung von SDK für Kotlin
- CloudWatch Protokolliert Beispiele mit dem SDK für Kotlin
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Comprehend unter Verwendung von SDK für Kotlin
- DynamoDB-Beispiele unter Verwendung von SDK für Kotlin
- EC2 Amazon-Beispiele mit SDK für Kotlin
- Beispiele für Amazon ECR unter Verwendung von SDK für Kotlin
- OpenSearch Servicebeispiele, die das SDK für Kotlin verwenden
- EventBridge Beispiele für die Verwendung von SDK für Kotlin
- AWS Glue Beispiele für die Verwendung von SDK für Kotlin

- IAM-Beispiele unter Verwendung von SDK für Kotlin
- AWS IoT Beispiele für die Verwendung von SDK für Kotlin
- AWS IoT data Beispiele für die Verwendung von SDK für Kotlin
- AWS IoT FleetWise Beispiele für die Verwendung von SDK für Kotlin
- Beispiele für Amazon Keyspaces unter Verwendung von SDK für Kotlin
- AWS KMS Beispiele für die Verwendung von SDK für Kotlin
- Lambda-Beispiele unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Location unter Verwendung von SDK für Kotlin
- MediaConvert Beispiele für die Verwendung von SDK für Kotlin
- Beispiele für Amazon Pinpoint unter Verwendung von SDK für Kotlin
- Beispiele für Amazon RDS unter Verwendung von SDK für Kotlin
- Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Redshift unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Rekognition unter Verwendung von SDK für Kotlin
- Beispiele für die Route-53-Domainregistrierung unter Verwendung von SDK für Kotlin
- Amazon-S3-Beispiele unter Verwendung von SDK für Kotlin
- SageMaker KI-Beispiele, die SDK für Kotlin verwenden
- Beispiele für Secrets Manager unter Verwendung von SDK für Kotlin
- Beispiele für Amazon SES unter Verwendung von SDK für Kotlin
- Beispiele für Amazon SNS unter Verwendung von SDK für Kotlin
- Beispiele für Amazon SQS unter Verwendung von SDK für Kotlin
- Beispiele für Step Functions unter Verwendung von SDK für Kotlin
- Support Beispiele für die Verwendung von SDK für Kotlin
- Beispiele für Amazon Translate unter Verwendung von SDKs für Kotlin

API-Gateway-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit API Gateway Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

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

Aurora-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aurora Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
```

```
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            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
 ${engineOb.engineVersion}")
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - DBClusterSchnappschuss erstellen
  - CreateDBInstance
  - LöschenDBCluster
  - LöschenDBClusterParameterGroup
  - LöschenDBInstance
  - Beschreiben DBCluster ParameterGroups
  - Beschreiben Sie die DBCluster Parameter
  - Beschreiben Sie DBCluster Schnappschüsse

- Beschreiben DBClusters
- DBEngineVersionen beschreiben
- Beschreiben DBInstances
- DescribeOrderableDBInstanceOptionen
- Modifizieren SieDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
```

```
        return response.dbCluster?.dbClusterArn
    }
}
```

- API-Details finden Sie unter API-Referenz DBCluster im AWS SDK für Kotlin erstellen.

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- API-Details finden Sie unter API-Referenz DBCluster ParameterGroup im AWS SDK für Kotlin erstellen.

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterSnapshot`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- API-Details finden Sie unter DBClusterSnapshot im AWS SDK erstellen für die Kotlin-API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- API-Details finden Sie unter API-Referenz DBInstance im AWS SDK für Kotlin erstellen.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- API-Details finden Sie unter Löschen DBCluster im AWS SDK für die Kotlin-API-Referenz.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
```

```
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { 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.")
    }
}
```

- API-Details finden Sie unter Löschen DBCluster ParameterGroup im AWS SDK für die Kotlin-API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- API-Details finden Sie unter Löschen DBInstance im AWS SDK für die Kotlin-API-Referenz.

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- Einzelheiten zur API finden Sie unter DBClusterParameterGroupsDescribe in AWS SDK for Kotlin API-Referenz.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- API-Details finden Sie unter Describe DBCluster Parameters in AWS SDK for Kotlin API-Referenz.

`DescribeDBClusterSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterSnapshots`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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!")
}
```

- Einzelheiten zur API finden Sie unter Beschreiben von DBCluster Snapshots im AWS SDK für die Kotlin-API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBClusters`

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie unter DBClustersDescribe in AWS SDK for Kotlin API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie unter DBEngineVersionen beschreiben im AWS SDK für die Kotlin-API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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")
}
```

- Einzelheiten zur API finden Sie unter DBInstancesDescribe in AWS SDK for Kotlin API-Referenz.

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- API-Details finden Sie unter Modify DBCluster ParameterGroup in AWS SDK for Kotlin API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Auto-Scaling-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon EC2 Auto Scaling Scaling-Gruppe mit einer Startvorlage und Availability Zones und erhalten Sie Informationen über laufende Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.numberOfAutoScalingGroups}")
    }
}

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")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateAutoScalingGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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!")
    }
}
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der API-Referenz zum AWS SDK für Kotlin.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
    }
}
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der API-Referenz zum AWS SDK für Kotlin.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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)
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der API-Referenz zum AWS SDK für Kotlin.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der API-Referenz zum AWS SDK für Kotlin.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der API-Referenz zum AWS SDK für Kotlin.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
```

```
    }
}
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der API-Referenz zum AWS SDK für Kotlin.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
    }
}
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der API-Referenz zum AWS SDK für Kotlin.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
    }
}
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der API-Referenz zum AWS SDK für Kotlin.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
    }
}
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der API-Referenz zum AWS SDK für Kotlin.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
    }
}
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon Bedrock unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung. `ListFoundationModels`

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Amazon-Bedrock-Basismodelle.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient.fromEnvironment { 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
    }
}
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit Amazon Bedrock Runtime verwenden.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Amazon Nova

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
suspend fun main() {
    converse().also { println(it) }
```

```
}

suspend fun converse(): String {
    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in one line."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Send the request and process the model's response
        runCatching {
            val response = client.converse(request)
            return response.output!!.asMessage().content.first().asText()
        }.getOrElse { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId",
 error)
        }
    }
}
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Kotlin.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamOutput
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message with a prompt
 * - Configure a streaming request with parameters
 * - Process the response stream in real time
 */
suspend fun main() {
    converseStream()
}

suspend fun converseStream(): String {
    // A buffer to collect the complete response
    val completeResponseBuffer = StringBuilder()

    // Create and configure the Bedrock runtime client
```

```
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in a
 paragraph."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseStreamRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Process the streaming response
        runCatching {
            client.converseStream(request) { response ->
                response.stream?.collect { chunk ->
                    when (chunk) {
                        is ConverseStreamOutput.ContentBlockDelta -> {
                            // Process each text chunk as it arrives
                            chunk.value.delta?.asText()?.let { text ->
                                print(text)
                                System.out.flush() // Ensure immediate output
                                completeResponseBuffer.append(text)
                            }
                        }
                        else -> {} // Other output block types can be handled as
 needed
                    }
                }
            }
```

```
        }.onFailure { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId:
 $error", error)
        }
    }

    return completeResponseBuffer.toString()
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der API-Referenz zum AWS SDK für Kotlin.

CloudWatch Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. CloudWatch

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo CloudWatch

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von CloudWatch beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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.fromEnvironment { 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}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListMetricsin der API-Referenz zum AWS SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Listet CloudWatch Namespaces und Metriken auf.
- Rufen Sie Statistiken für eine Metrik und die geschätzte Fakturierung ab.
- Erstellen und aktualisieren Sie ein Dashboard.
- Erstellen Sie eine Metrik und fügen Sie ihr Daten hinzu.
- Erstellen und lösen Sie einen Alarm aus und zeigen Sie dann den Alarmverlauf an.
- Fügen Sie einen Anomaliedetektor hinzu.
- Ermitteln Sie ein Metrik-Image, dann bereinigen Sie die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem CloudWatch Funktionen demonstriert werden.

```
/**
 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 inOb = 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 = inOb.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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard

- PutMetricAlarm
- PutMetricData

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlarms`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- Einzelheiten zur API finden Sie DeleteAlarmsin der API-Referenz zum AWS SDK für Kotlin.

`DeleteAnomalyDetector`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAnomalyDetector`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteAnomalyDetectorin der API-Referenz zum AWS SDK für Kotlin.

`DeleteDashboards`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDashboards`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteDashboardsin der API-Referenz zum AWS SDK für Kotlin.

`DescribeAlarmHistory`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmHistory`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAlarmHistoryin der API-Referenz zum AWS SDK für Kotlin.

`DescribeAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarms`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DescribeAlarmsin der API-Referenz zum AWS SDK für Kotlin.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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.")
        }
    }
}
```

- API-Details finden Sie DescribeAlarmsForMetricin der API-Referenz zum AWS SDK für Kotlin.

`DescribeAnomalyDetectors`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAnomalyDetectors`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DescribeAnomalyDetectorsin der API-Referenz zum AWS SDK für Kotlin.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- API-Details finden Sie DisableAlarmActionsin der API-Referenz zum AWS SDK für Kotlin.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- API-Details finden Sie EnableAlarmActionsin der API-Referenz zum AWS SDK für Kotlin.

`GetMetricData`

Das folgende Codebeispiel zeigt die VerwendungGetMetricData.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie GetMetricDatain der API-Referenz zum AWS SDK für Kotlin.

`GetMetricStatistics`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricStatistics`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
            }
        }
    }
}
```

- API-Details finden Sie GetMetricStatisticsin der API-Referenz zum AWS SDK für Kotlin.

`GetMetricWidgetImage`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricWidgetImage`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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")
}
```

- API-Details finden Sie GetMetricWidgetImagein der API-Referenz zum AWS SDK für Kotlin.

`ListDashboards`

Das folgende Codebeispiel zeigt die Verwendung`ListDashboards`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}")
            }
    }
```

```
}
```

- API-Details finden Sie ListDashboardsin der API-Referenz zum AWS SDK für Kotlin.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { 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
}
```

- API-Details finden Sie ListMetricsin der API-Referenz zum AWS SDK für Kotlin.

`PutAnomalyDetector`

Das folgende Codebeispiel zeigt die Verwendung`PutAnomalyDetector`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- API-Details finden Sie PutAnomalyDetectorin der API-Referenz zum AWS SDK für Kotlin.

`PutDashboard`

Das folgende Codebeispiel zeigt die Verwendung`PutDashboard`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- API-Details finden Sie PutDashboardin der API-Referenz zum AWS SDK für Kotlin.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun putMetricAlarm(
    alarmNameVal: String,
    instanceIdVal: String,
) {
    val dimensionOb =
        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(dimensionOb)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- API-Details finden Sie PutMetricAlarmin der API-Referenz zum AWS SDK für Kotlin.

`PutMetricData`

Das folgende Codebeispiel zeigt die VerwendungPutMetricData.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- API-Details finden Sie PutMetricDatain der API-Referenz zum AWS SDK für Kotlin.

CloudWatch Protokolliert Beispiele mit dem SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit CloudWatch Logs Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSubscriptionFilter`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- API-Details finden Sie DeleteSubscriptionFilterin der API-Referenz zum AWS SDK für Kotlin.

`DescribeSubscriptionFilters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubscriptionFilters`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DescribeSubscriptionFiltersin der API-Referenz zum AWS SDK für Kotlin.

`StartLiveTail`

Das folgende Codebeispiel zeigt die Verwendung`StartLiveTail`.

SDK für Kotlin

Binden Sie die erforderlichen Dateien ein.

```
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
```

Starten Sie die Live-Tail-Sitzung.

```
    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 unsubcribe 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)
    }
```

- Einzelheiten zur API finden Sie StartLiveTailin der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit Amazon Cognito Identity Provider verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung. `AdminGetUser`

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
```

```
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- API-Details finden Sie AdminGetUserin der API-Referenz zum AWS SDK für Kotlin.

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die VerwendungAdminInitiateAuth.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- API-Details finden Sie AdminInitiateAuthin der API-Referenz zum AWS SDK für Kotlin.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die VerwendungAdminRespondToAuthChallenge.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
```

```
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- API-Details finden Sie AdminRespondToAuthChallengein der API-Referenz zum AWS SDK für Kotlin.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { 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
    }
}
```

- API-Details finden Sie AssociateSoftwareTokenin der API-Referenz zum AWS SDK für Kotlin.

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSignUp`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- API-Details finden Sie ConfirmSignUpin der API-Referenz zum AWS SDK für Kotlin.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- API-Details finden Sie ListUsersin der API-Referenz zum AWS SDK für Kotlin.

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung`ResendConfirmationCode`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- API-Details finden Sie ResendConfirmationCodein der API-Referenz zum AWS SDK für Kotlin.

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung`SignUp`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API-Details finden Sie SignUpin der API-Referenz zum AWS SDK für Kotlin.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- API-Details finden Sie VerifySoftwareTokenin der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.
- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.

- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 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 inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a new
code? (Yes/No)")
    val ans = inOb.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 = inOb.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 = inOb.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 = inOb.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.fromEnvironment { 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.fromEnvironment { 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 challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Beispiele für Amazon Comprehend unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Comprehend Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Nachrichtenanwendung

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS eine Nachrichtenanwendung erstellen können.

SDK für Kotlin

> Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.
>
> Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub
>
> In diesem Beispiel verwendete Dienste
>
> - Amazon Comprehend
> - Amazon SQS

DynamoDB-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine DynamoDB-Tabelle.

```
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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { 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}")
    }
}
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
// 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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

Rufen Sie ein Element aus einer Tabelle ab.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

Vollständiges Beispiel.

```
suspend fun main() {
    val tableName = "Movies"
    val fileName = "../../../resources/sample_files/movies.json"
    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.")
    deletIssuesTable(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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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]}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`CreateTable`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateTable`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { 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
```

```
    }
}
```

- API-Details finden Sie CreateTablein der API-Referenz zum AWS SDK für Kotlin.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- API-Details finden Sie DeleteItemin der API-Referenz zum AWS SDK für Kotlin.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- API-Details finden Sie DeleteTablein der API-Referenz zum AWS SDK für Kotlin.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- API-Details finden Sie GetItemin der API-Referenz zum AWS SDK für Kotlin.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllTables() {
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
```

```
            println("Table name is $tableName")
        }
    }
}
```

- API-Details finden Sie ListTablesin der API-Referenz zum AWS SDK für Kotlin.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- API-Details finden Sie PutItemin der API-Referenz zum AWS SDK für Kotlin.

`Query`

Das folgende Codebeispiel zeigt die VerwendungQuery.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der API–Referenz zum AWS SDK für Kotlin.

`Scan`

Das folgende Codebeispiel zeigt, wie man es benutzt`Scan`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { 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]}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz zum AWS -SDK für Kotlin.

`UpdateItem`

Das folgende Codebeispiel zeigt, wie man es benutzt`UpdateItem`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- API-Details finden Sie UpdateItemin der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

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

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
```

```
    val tableName = "MoviesPartiQBatch"
    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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
```

```
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,
 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
```

```
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
```

```
        }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
```

```
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- API-Details finden Sie BatchExecuteStatementin der API-Referenz zum AWS SDK für Kotlin.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQ"
    val fileName = "../../../resources/sample_files/movies.json"
    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 request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            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)
}
```

- API-Details finden Sie ExecuteStatementin der API-Referenz zum AWS SDK für Kotlin.

EC2 Amazon-Beispiele mit SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit Amazon EC2 verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen

- Aktionen

Erste Schritte

Hallo Amazon EC2

Das folgende Codebeispiel zeigt, wie Sie mit Amazon beginnen können EC2.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitUntilInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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")
    }
}

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt, wie man es benutzt`AllocateAddress`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- API-Details finden Sie AllocateAddressin der API-Referenz zum AWS SDK für Kotlin.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- API-Details finden Sie AssociateAddressin der API-Referenz zum AWS SDK für Kotlin.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2SecurityGroupSc(
```

```
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { 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
    }
```

```
}
```

- API-Details finden Sie AuthorizeSecurityGroupIngressin der API-Referenz zum AWS SDK für Kotlin.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- API-Details finden Sie CreateKeyPairin der API-Referenz zum AWS SDK für Kotlin.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- API-Details finden Sie CreateSecurityGroupin der API-Referenz zum AWS SDK für Kotlin.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- API-Details finden Sie DeleteKeyPairin der API-Referenz zum AWS SDK für Kotlin.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- API-Details finden Sie DeleteSecurityGroupin der API-Referenz zum AWS SDK für Kotlin.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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.fromEnvironment { 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
    }
}
```

- API-Details finden Sie DescribeInstanceTypesin der API-Referenz zum AWS SDK für Kotlin.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client.fromEnvironment { 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}")
            }
        }
    }
}
```

- API-Details finden Sie DescribeInstancesin der API-Referenz zum AWS SDK für Kotlin.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2Keys() {
    Ec2Client.fromEnvironment { 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}")
```

```
        }
    }
}
```

- API-Details finden Sie DescribeKeyPairsin der API-Referenz zum AWS SDK für Kotlin.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für Kotlin.

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateAddress`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- API-Details finden Sie DisassociateAddressin der API-Referenz zum AWS SDK für Kotlin.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }
```

```
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- API-Details finden Sie ReleaseAddressin der API-Referenz zum AWS SDK für Kotlin.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- API-Details finden Sie RunInstancesin der API-Referenz zum AWS SDK für Kotlin.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { 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")
    }
```

```
}
```

- API-Details finden Sie StartInstancesin der API-Referenz zum AWS SDK für Kotlin.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { 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")
    }
}
```

- API-Details finden Sie StopInstancesin der API-Referenz zum AWS SDK für Kotlin.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie TerminateInstancesin der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon ECR unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon ECR Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon ECR

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon ECR.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS -SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-ECR-Repository.
- Legen Sie Repository-Richtlinien fest.
- Repository abrufen URIs.
- Rufen Sie Amazon-ECR-Autorisierungs-Token ab.
- Legen Sie Richtlinien für den Lebenszyklus von Amazon-ECR-Repositorys fest.
- Pushen Sie ein Docker-Image in ein Amazon-ECR-Repository.
- Verifizieren Sie das Vorhandensein eines Images in einem Amazon-ECR-Repository.
- Listen Sie die Amazon-ECR-Repositorys für Ihr Konto auf und erhalten Sie detaillierte Informationen dazu.
- Löschen Sie Amazon-ECR-Repositorys.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Amazon-ECR-Funktionen demonstriert.

```
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:
 *
 * /scenarios/basics/ecr/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 = args[0]
   var localImageName: String
   var accountId = args[1]
   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.")
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon ECR SDK.

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy

- StartLifecyclePolicyPreview

Aktionen

`CreateRepository`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateRepository`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.fromEnvironment { 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
        }
    }
```

- API-Details finden Sie CreateRepositoryin der API-Referenz zum AWS SDK für Kotlin.

`DeleteRepository`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRepository`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
```

```
        }
    }
```

- API-Details finden Sie DeleteRepositoryin der API-Referenz zum AWS SDK für Kotlin.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.fromEnvironment { 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.")
            }
        }
    }
```

- API-Details finden Sie DescribeImagesin der API-Referenz zum AWS SDK für Kotlin.

`DescribeRepositories`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRepositories`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.fromEnvironment { 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 ""
            }
        }
    }
```

- API-Details finden Sie DescribeRepositoriesin der API-Referenz zum AWS SDK für Kotlin.

`GetAuthorizationToken`

Das folgende Codebeispiel zeigt die Verwendung`GetAuthorizationToken`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { 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.")
            }
        }
    }
```

- API-Details finden Sie GetAuthorizationTokenin der API-Referenz zum AWS SDK für Kotlin.

`GetRepositoryPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetRepositoryPolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- API-Details finden Sie GetRepositoryPolicyin der API-Referenz zum AWS SDK für Kotlin.

`PushImageCmd`

Das folgende Codebeispiel zeigt die Verwendung`PushImageCmd`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.fromEnvironment { 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)
            }
        }
    }
```

- API-Details finden Sie PushImageCmdin der API-Referenz zum AWS SDK für Kotlin.

`SetRepositoryPolicy`

Das folgende Codebeispiel zeigt die Verwendung`SetRepositoryPolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- API-Details finden Sie SetRepositoryPolicyin der API-Referenz zum AWS SDK für Kotlin.

`StartLifecyclePolicyPreview`

Das folgende Codebeispiel zeigt die Verwendung`StartLifecyclePolicyPreview`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.fromEnvironment { 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.")
```

```
            }
        }
    }
```

- API-Details finden Sie StartLifecyclePolicyPreviewin der API-Referenz zum AWS SDK für Kotlin.

OpenSearch Servicebeispiele, die das SDK für Kotlin verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit OpenSearch Service Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDomain`

Das folgende Codebeispiel zeigt die Verwendung`CreateDomain`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewDomain(domainNameVal: String?) {
```

```
    val clusterConfigOb =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }

    val ebsOptionsOb =
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptionsOb =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
            clusterConfig = clusterConfigOb
            ebsOptions = ebsOptionsOb
            nodeToNodeEncryptionOptions = encryptionOptionsOb
        }

    println("Sending domain creation request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- API-Details finden Sie CreateDomainin der API-Referenz zum AWS SDK für Kotlin.

`DeleteDomain`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDomain`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteDomainin der API-Referenz zum AWS SDK für Kotlin.

`ListDomainNames`

Das folgende Codebeispiel zeigt die Verwendung`ListDomainNames`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllDomains() {
```

```
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- API-Details finden Sie ListDomainNamesin der API-Referenz zum AWS SDK für Kotlin.

`UpdateDomainConfig`

Das folgende Codebeispiel zeigt die Verwendung`UpdateDomainConfig`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
```

```
}
```

- API-Details finden Sie UpdateDomainConfigin der API-Referenz zum AWS SDK für Kotlin.

EventBridge Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. EventBridge

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo EventBridge

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von EventBridge beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie ListEventBusesin der API-Referenz zum AWS SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Regel und fügen Sie ihr ein Ziel hinzu.
- Aktivieren und deaktivieren Sie Regeln.
- Listen Sie Regeln und Ziele auf und aktualisieren Sie sie.
- Senden Sie Ereignisse und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
```

```
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
```

```
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
```

```
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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"
            }
        ]
    }
```

```
    """.trimIndent()

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient.fromEnvironment { 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.fromEnvironment { 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"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Aktionen

`DeleteRule`

Das folgende Codebeispiel zeigt, wie man es benutzt`DeleteRule`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- API-Details finden Sie DeleteRulein der API-Referenz zum AWS SDK für Kotlin.

`DescribeRule`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRule`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
```

```
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- API-Details finden Sie DescribeRulein der API-Referenz zum AWS SDK für Kotlin.

`DisableRule`

Das folgende Codebeispiel zeigt die Verwendung`DisableRule`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- API-Details finden Sie DisableRulein der API-Referenz zum AWS SDK für Kotlin.

`EnableRule`

Das folgende Codebeispiel zeigt die Verwendung`EnableRule`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- API-Details finden Sie EnableRulein der API-Referenz zum AWS SDK für Kotlin.

`ListRuleNamesByTarget`

Das folgende Codebeispiel zeigt die Verwendung`ListRuleNamesByTarget`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- API-Details finden Sie ListRuleNamesByTargetin der API-Referenz zum AWS SDK für Kotlin.

`ListRules`

Das folgende Codebeispiel zeigt die Verwendung`ListRules`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie ListRulesin der API-Referenz zum AWS SDK für Kotlin.

`ListTargetsByRule`

Das folgende Codebeispiel zeigt die Verwendung`ListTargetsByRule`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- API-Details finden Sie ListTargetsByRulein der API-Referenz zum AWS SDK für Kotlin.

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- API-Details finden Sie PutEventsin der API-Referenz zum AWS SDK für Kotlin.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine geplante Regel.

```
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.fromEnvironment { region = "us-west-2" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

Erstellen Sie eine Regel, die ausgelöst wird, wenn ein Objekt zu einem Amazon-Simple-Storage-Service-Bucket hinzugefügt wird.

```
// 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"]
            }
        }
    }
    """.trimIndent()
```

```
    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- API-Details finden Sie PutRulein der API-Referenz zum AWS SDK für Kotlin.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
```

```
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}
```

Fügen Sie einen Eingabe-Transformator als Ziel für eine Regel hinzu.

```
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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- API-Details finden Sie PutTargetsin der API-Referenz zum AWS SDK für Kotlin.

`RemoveTargets`

Das folgende Codebeispiel zeigt die Verwendung`RemoveTargets`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { 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")
            }
        }
    }
}
```

- API-Details finden Sie RemoveTargetsin der API-Referenz zum AWS SDK für Kotlin.

AWS Glue Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. AWS Glue

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon-S3-Bucket crawlt und eine Datenbank mit CSV-formatierten Metadaten generiert.
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem AWS Glue Data Catalog auf.
- Erstellen Sie einen Auftrag, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten umzuwandeln und die JSON-formatierte Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für Kotlin

> **Note**
>
> Es gibt mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
            maxResults = 10
        }

    GlueClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase

- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateCrawler`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- API-Details finden Sie CreateCrawlerin der API-Referenz zum AWS SDK für Kotlin.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
```

```
    GlueClient.fromEnvironment { 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")
    }
}
```

- API-Details finden Sie GetCrawlerin der API-Referenz zum AWS SDK für Kotlin.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- API-Details finden Sie GetDatabasein der API-Referenz zum AWS SDK für Kotlin.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- API-Details finden Sie StartCrawlerin der API-Referenz zum AWS SDK für Kotlin.

IAM-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit IAM Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen

- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Benutzer-Aktionen umschließen.

```
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 = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:*"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment {
        region = "us-east-1"
        credentialsProvider = staticCredentials
    }

    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.fromEnvironment { 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.fromEnvironment { 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)
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole

- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt, wie man es benutzt`AttachRolePolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { 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
}
```

- API-Details finden Sie AttachRolePolicyin der API-Referenz zum AWS SDK für Kotlin.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMAccessKey(user: String?): String {
```

```
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- API-Details finden Sie CreateAccessKeyin der API-Referenz zum AWS SDK für Kotlin.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- API-Details finden Sie CreateAccountAliasin der API-Referenz zum AWS SDK für Kotlin.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- API-Details finden Sie CreatePolicyin der API-Referenz zum AWS SDK für Kotlin.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- API-Details finden Sie CreateUserin der API-Referenz zum AWS SDK für Kotlin.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- API-Details finden Sie DeleteAccessKeyin der API-Referenz zum AWS SDK für Kotlin.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIAMAccountAlias(alias: String) {
```

```
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- API-Details finden Sie DeleteAccountAliasin der API-Referenz zum AWS SDK für Kotlin.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- API-Details finden Sie DeletePolicyin der API-Referenz zum AWS SDK für Kotlin.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- API-Details finden Sie DeleteUserin der API-Referenz zum AWS SDK für Kotlin.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- API-Details finden Sie DetachRolePolicyin der API-Referenz zum AWS SDK für Kotlin.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
```

```
}
```

- API-Details finden Sie GetPolicyin der API-Referenz zum AWS SDK für Kotlin.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- API-Details finden Sie ListAccessKeysin der API-Referenz zum AWS SDK für Kotlin.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAliases() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- API-Details finden Sie ListAccountAliasesin der API-Referenz zum AWS SDK für Kotlin.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllUsers() {
    IamClient.fromEnvironment { 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}")
            }
        }
    }
}
```

- API-Details finden Sie ListUsersin der API-Referenz zum AWS SDK für Kotlin.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- API-Details finden Sie UpdateUserin der API-Referenz zum AWS SDK für Kotlin.

AWS IoT Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. AWS IoT

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listThings in der API-Referenz zum AWS -SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erschaffe ein AWS IoT Ding.
- Generieren Sie ein Gerätezertifikat.
- Aktualisiere ein AWS IoT Ding mit Attributen.
- Geben Sie einen eindeutigen Endpunkt zurück.
- Listen Sie Ihre AWS IoT Zertifikate auf.
- Aktualisiere einen AWS IoT Schatten.
- Schreiben Sie Statusinformationen aus.
- Erstellt eine Regel.

- Listen Sie Ihre Regeln auf.
- Suchen Sie Objekte anhand des Objektnamens.
- Lösche ein AWS IoT Ding.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development environment
 * is set up, including configuring your credentials.
 *
```

```
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-
started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
```

```
        This example program demonstrates various interactions with the AWS Internet
of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an IoT
thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that can
be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
        A device certificate performs a role in securing the communication between
devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
```

```
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
```

```
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example, you
can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
```

```
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName? (y/
n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
 delete.")
    }
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
```

```
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase = true))
 {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

Aktionen

`AttachThingPrincipal`

Das folgende Codebeispiel zeigt, wie man es benutzt`AttachThingPrincipal`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
```

```
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- API-Details finden Sie AttachThingPrincipalin der API-Referenz zum AWS SDK für Kotlin.

`CreateKeysAndCertificate`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeysAndCertificate`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { 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
```

```
    }
}
```

- API-Details finden Sie CreateKeysAndCertificatein der API-Referenz zum AWS SDK für Kotlin.

`CreateThing`

Das folgende Codebeispiel zeigt die Verwendung`CreateThing`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- API-Details finden Sie CreateThingin der API-Referenz zum AWS SDK für Kotlin.

`CreateTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopicRule`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- API-Details finden Sie CreateTopicRulein der API-Referenz zum AWS SDK für Kotlin.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteCertificatein der API-Referenz zum AWS SDK für Kotlin.

`DeleteThing`

Das folgende Codebeispiel zeigt die Verwendung`DeleteThing`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- API-Details finden Sie DeleteThingin der API-Referenz zum AWS SDK für Kotlin.

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DescribeEndpoint`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { 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
    }
}
```

- API-Details finden Sie DescribeEndpointin der API-Referenz zum AWS SDK für Kotlin.

`DescribeThing`

Das folgende Codebeispiel zeigt die Verwendung`DescribeThing`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- API-Details finden Sie DescribeThingin der API-Referenz zum AWS SDK für Kotlin.

`DetachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`DetachThingPrincipal`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- API-Details finden Sie DetachThingPrincipalin der API-Referenz zum AWS SDK für Kotlin.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie ListCertificatesin der API-Referenz zum AWS SDK für Kotlin.

`SearchIndex`

Das folgende Codebeispiel zeigt die Verwendung`SearchIndex`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { 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}") }
        }
    }
}
```

- API-Details finden Sie SearchIndexin der API-Referenz zum AWS SDK für Kotlin.

`UpdateThing`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThing`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- API-Details finden Sie UpdateThingin der API-Referenz zum AWS SDK für Kotlin.

AWS IoT data Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. AWS IoT data

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`GetThingShadow`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { 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")
    }
}
```

- API-Details finden Sie GetThingShadowin der API-Referenz zum AWS SDK für Kotlin.

`UpdateThingShadow`

Das folgende Codebeispiel zeigt die VerwendungUpdateThingShadow.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- API-Details finden Sie UpdateThingShadowin der API-Referenz zum AWS SDK für Kotlin.

AWS IoT FleetWise Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. AWS IoT FleetWise

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT FleetWise

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT FleetWise beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
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() {
    listSignalCatalogs()
}

/**
 * Lists the AWS FleetWise Signal Catalogs associated with the current AWS account.
 */
suspend fun listSignalCatalogs() {
    val request = ListSignalCatalogsRequest {
        maxResults = 10
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogs(request)
        val summaries = response.summaries

        if (summaries.isNullOrEmpty()) {
            println("No AWS FleetWise Signal Catalogs were found.")
        } else {
            summaries.forEach { summary ->
                with(summary) {
                    println("Catalog Name: $name")
                    println("ARN: $arn")
                    println("Created: $creationTime")
                    println("Last Modified: $lastModificationTime")
                    println("---------------")
                }
            }
        }
    }
}
```

- API-Details finden Sie unter listSignalCatalogsPaginator im AWS SDK für die Kotlin-API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Sammlung standardisierter Signale.
- Erstellen Sie eine Flotte, die eine Gruppe von Fahrzeugen darstellt.
- Erstellen Sie ein Modellmanifest.
- Erstellen Sie ein Decoder-Manifest.
- Überprüfen Sie den Status des Modellmanifests.
- Überprüfen Sie den Status des Decoders.
- Erstellen Sie ein IoT-Objekt.
- Erstellen Sie ein Fahrzeug.
- Zeigen Sie Fahrzeugdetails an.
- Löschen Sie die Assets. AWS IoT FleetWise

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS IoT SiteWise Funktionen demonstriert werden.

```
/**
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
 */
var scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
```

```
    val usage =
        """
        Usage:
            <signalCatalogName> <manifestName> <fleetId> <vecName> <decName>

        Where:
            signalCatalogName     - The name of the Signal Catalog to create (eg,
catalog30).
            manifestName          - The name of the Vehicle Model (Model Manifest)
to create (eg, manifest30).
            fleetId               - The ID of the Fleet to create (eg, fleet30).
            vecName               - The name of the Vehicle to create (eg,
vehicle30).
            decName               - The name of the Decoder Manifest to create (eg,
decManifest30).

        """.trimIndent()

    if (args.size != 5) {
        println(usage)
        return
    }

    val signalCatalogName = args[0]
    val manifestName = args[1]
    val fleetId = args[2]
    val vecName = args[3]
    val decName = args[4]

    println(
        """
        AWS IoT FleetWise is a managed service that simplifies the
        process of collecting, organizing, and transmitting vehicle
        data to the cloud in near real-time. Designed for automakers
        and fleet operators, it allows you to define vehicle models,
        specify the exact data you want to collect (such as engine
        temperature, speed, or battery status), and send this data to
        AWS for analysis. By using intelligent data collection
        techniques, IoT FleetWise reduces the volume of data
        transmitted by filtering and transforming it at the edge,
        helping to minimize bandwidth usage and costs.

        At its core, AWS IoT FleetWise helps organizations build
        scalable systems for vehicle data management and analytics,
```

```
        supporting a wide variety of vehicles and sensor configurations.
        You can define signal catalogs and decoder manifests that describe
        how raw CAN bus signals are translated into readable data, making
        the platform highly flexible and extensible. This allows
        manufacturers to optimize vehicle performance, improve safety,
        and reduce maintenance costs by gaining real-time visibility
        into fleet operations.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    println(DASHES)
    runScenario(signalCatalogName, fleetId, manifestName, decName, vecName)
}

suspend fun runScenario(signalCatalogName: String, fleetIdVal: String, manifestName:
 String, decName: String, vecName: String) {
    println(DASHES)
    println("1. Creates a collection of standardized signals that can be reused to
 create vehicle models")
    waitForInputToContinue(scanner)
    val signalCatalogArn = createbranchVehicle(signalCatalogName)
    println("The collection ARN is $signalCatalogArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("2. Create a fleet that represents a group of vehicles")
    println(
        """
        Creating an IoT FleetWise fleet allows you to efficiently collect,
        organize, and transfer vehicle data to the cloud, enabling real-time
        insights into vehicle performance and health.

        It helps reduce data costs by allowing you to filter and prioritize
        only the most relevant vehicle signals, supporting advanced analytics
        and predictive maintenance use cases.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val fleetid = createFleet(signalCatalogArn, fleetIdVal)
    println("The fleet Id is $fleetid")
    waitForInputToContinue(scanner)
    val nodeList = listSignalCatalogNode(signalCatalogName)
    println(DASHES)
```

```
    println(DASHES)
    println("3. Create a model manifest")
    println(
        """
        An AWS IoT FleetWise manifest defines the structure and
        relationships of vehicle data. The model manifest specifies
        which signals to collect and how they relate to vehicle systems,
        while the decoder manifest defines how to decode raw vehicle data
        into meaningful signals.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val nodes = listSignalCatalogNode(signalCatalogName)
    val manifestArn = nodes?.let { createModelManifest(manifestName,
signalCatalogArn, it) }
    println("The manifest ARN is $manifestArn")
    println(DASHES)

    println(DASHES)
    println("4. Create a decoder manifest")
    println(
        """
        A decoder manifest in AWS IoT FleetWise defines how raw vehicle
        data (such as CAN signals) should be interpreted and decoded
        into meaningful signals. It acts as a translation layer
        that maps vehicle-specific protocols to standardized data formats
        using decoding rules. This is crucial for extracting usable
        data from different vehicle models, even when their data
        formats vary.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val decArn = createDecoderManifest(decName, manifestArn)
    println("The decoder manifest ARN is $decArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("5. Check the status of the model manifest")
    println(
        """
        The model manifest must be in an ACTIVE state before it can be used
        to create or update a vehicle.
```

```
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    updateModelManifest(manifestName)
    waitForModelManifestActive(manifestName)
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Check the status of the decoder")
    println(
        """
        The decoder manifest must be in an ACTIVE state before it can be used
        to create or update a vehicle.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    updateDecoderManifest(decName)
    waitForDecoderManifestActive(decName)
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Create an IoT Thing")
    println(
        """
        AWS IoT FleetWise expects an existing AWS IoT Thing with the same
        name as the vehicle name you are passing to createVehicle method.
        Before calling createVehicle(), you must create an AWS IoT Thing
        with the same name using the AWS IoT Core service.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    createThingIfNotExist(vecName)
    println(DASHES)

    println(DASHES)
    println("8. Create a vehicle")
    println(
        """
        Creating a vehicle in AWS IoT FleetWise allows you to digitally
        represent and manage a physical vehicle within the AWS ecosystem.
        This enables efficient ingestion, transformation, and transmission
        of vehicle telemetry data to the cloud for analysis.
```

```
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    createVehicle(vecName, manifestArn, decArn)
    println(DASHES)

    println(DASHES)
    println("9. Display vehicle details")
    waitForInputToContinue(scanner)
    getVehicleDetails(vecName)
    waitForInputToContinue(scanner)
    println(DASHES)
    println(DASHES)
    println("10. Delete the AWS IoT Fleetwise Assets")
    println("Would you like to delete the IoT Fleetwise Assets? (y/n)")
    val delAns = scanner.nextLine().trim()
    if (delAns.equals("y", ignoreCase = true)) {
        deleteVehicle(vecName)
        deleteDecoderManifest(decName)
        deleteModelManifest(manifestName)
        deleteFleet(fleetid)
        deleteSignalCatalog(signalCatalogName)
    }

    println(DASHES)
    println(
        """
        Thank you for checking out the AWS IoT Fleetwise Service Use demo. We hope
 you
        learned something new, or got some inspiration for your own apps today.
        For more AWS code examples, have a look at:
        https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-library.html
        """.trimIndent(),
    )
    println(DASHES)
}

suspend fun deleteVehicle(vecName: String) {
    val request = DeleteVehicleRequest {
        vehicleName = vecName
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
```

```
        fleetwiseClient.deleteVehicle(request)
        println("Vehicle $vecName was deleted successfully.")
    }
}

suspend fun getVehicleDetails(vehicleNameVal: String) {
    val request = GetVehicleRequest {
        vehicleName = vehicleNameVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.getVehicle(request)
        val details = mapOf(
            "vehicleName" to response.vehicleName,
            "arn" to response.arn,
            "modelManifestArn" to response.modelManifestArn,
            "decoderManifestArn" to response.decoderManifestArn,
            "attributes" to response.attributes.toString(),
            "creationTime" to response.creationTime.toString(),
            "lastModificationTime" to response.lastModificationTime.toString(),
        )

        println("Vehicle Details:")
        for ((key, value) in details) {
            println("• %-20s : %s".format(key, value))
        }
    }
}

suspend fun createVehicle(vecName: String, manifestArn: String?, decArn: String) {
    val request = CreateVehicleRequest {
        vehicleName = vecName
        modelManifestArn = manifestArn
        decoderManifestArn = decArn
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.createVehicle(request)
        println("Vehicle $vecName was created successfully.")
    }
}
```

```
/**
 * Creates an IoT Thing if it does not already exist.
 *
 * @param vecName the name of the IoT Thing to create
 */
suspend fun createThingIfNotExist(vecName: String) {
    val request = CreateThingRequest {
        thingName = vecName
    }

    IotClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.createThing(request)
        println("The $vecName IoT Thing was successfully created")
    }
}

suspend fun updateDecoderManifest(nameVal: String) {
    val request = UpdateDecoderManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.updateDecoderManifest(request)
        println("$nameVal was successfully updated")
    }
}

/**
 * Waits for the specified model manifest to become active.
 *
 * @param decNameVal the name of the model manifest to wait for
 */
suspend fun waitForDecoderManifestActive(decNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
```

```
                val request = GetDecoderManifestRequest {
                    name = decNameVal
                }

                val response = fleetwiseClient.getDecoderManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: INVALID #\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}

/**
 * Waits for the specified model manifest to become active.
 *
 * @param manifestName the name of the model manifest to wait for
 */
suspend fun waitForModelManifestActive(manifestNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        while (true) {
```

```
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetModelManifestRequest {
                    name = manifestNameVal
                }

                val response = fleetwiseClient.getModelManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: INVALID #
\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}

/**
 * Updates the model manifest.
 *
 * @param nameVal the name of the model manifest to update
 */
suspend fun updateModelManifest(nameVal: String) {
    val request = UpdateModelManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
```

```
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.updateModelManifest(request)
        println("$nameVal was successfully updated")
    }
}

suspend fun deleteDecoderManifest(nameVal: String) {
    val request = DeleteDecoderManifestRequest {
        name = nameVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteDecoderManifest(request)
        println("$nameVal was successfully deleted")
    }
}

/**
 * Creates a new decoder manifest.
 *
 * @param decName             the name of the decoder manifest
 * @param modelManifestArnVal the ARN of the model manifest
 * @return the ARN of the decoder manifest
 */
suspend fun createDecoderManifest(decName: String, modelManifestArnVal: String?):
 String {
    val interfaceIdVal = "can0"

    val canInter = CanInterface {
        name = "canInterface0"
        protocolName = "CAN"
        protocolVersion = "1.0"
    }

    val networkInterface = NetworkInterface {
        interfaceId = interfaceIdVal
        type = NetworkInterfaceType.CanInterface
        canInterface = canInter
    }

    val carRpmSig = CanSignal {
```

```
        messageId = 100
        isBigEndian = false
        isSigned = false
        startBit = 16
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val carSpeedSig = CanSignal {
        messageId = 101
        isBigEndian = false
        isSigned = false
        startBit = 0
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val engineRpmDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carRpmSig
    }

    val vehicleSpeedDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carSpeedSig
    }

    val request = CreateDecoderManifestRequest {
        name = decName
        modelManifestArn = modelManifestArnVal
        networkInterfaces = listOf(networkInterface)
        signalDecoders = listOf(engineRpmDecoder, vehicleSpeedDecoder)
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
{ fleetwiseClient ->
        val response = fleetwiseClient.createDecoderManifest(request)
        return response.arn
```

```
    }
}

/**
 * Deletes a signal catalog.
 *
 * @param name the name of the signal catalog to delete
 */
suspend fun deleteSignalCatalog(catName: String) {
    val request = DeleteSignalCatalogRequest {
        name = catName
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteSignalCatalog(request)
        println(" $catName was successfully deleted")
    }
}

/**
 * Deletes a fleet based on the provided fleet ID.
 *
 * @param fleetId the ID of the fleet to be deleted
 */
suspend fun deleteFleet(fleetIdVal: String) {
    val request = DeleteFleetRequest {
        fleetId = fleetIdVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteFleet(request)
        println(" $fleetIdVal was successfully deleted")
    }
}

/**
 * Deletes a model manifest.
 *
 * @param nameVal the name of the model manifest to delete
 */
suspend fun deleteModelManifest(nameVal: String) {
    val request = DeleteModelManifestRequest {
        name = nameVal
```

```
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteModelManifest(request)
        println(" $nameVal was successfully deleted")
    }
}

/**
 * Creates a model manifest.
 *
 * @param name              the name of the model manifest to create
 * @param signalCatalogArn  the Amazon Resource Name (ARN) of the signal catalog
 * @param nodes             a list of nodes to include in the model manifest
 * @return a {@link CompletableFuture} that completes with the ARN of the created
 model manifest
 */
suspend fun createModelManifest(nameVal: String, signalCatalogArnVal: String,
 nodesList: List<Node>): String {
    val fqnList: List<String> = nodesList.map { node ->
        when (node) {
            is Node.Sensor -> node.asSensor().fullyQualifiedName
            is Node.Branch -> node.asBranch().fullyQualifiedName
            else -> throw RuntimeException("Unsupported node type")
        }
    }

    val request = CreateModelManifestRequest {
        name = nameVal
        signalCatalogArn = signalCatalogArnVal
        nodes = fqnList
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createModelManifest(request)
        return response.arn
    }
}

/**
 * Lists the signal catalog nodes asynchronously.
 *
 * @param signalCatalogName the name of the signal catalog
```

```
 * @return a CompletableFuture that, when completed, contains a list of nodes in the
 specified signal catalog
 * @throws CompletionException if an exception occurs during the asynchronous
 operation
 */
suspend fun listSignalCatalogNode(signalCatalogName: String): List<Node>? {
    val request = ListSignalCatalogNodesRequest {
        name = signalCatalogName
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogNodes(request)
        return response.nodes
    }
}

/**
 * Creates a new fleet.
 *
 * @param catARN the Amazon Resource Name (ARN) of the signal catalog to associate
 with the fleet
 * @param fleetId the unique identifier for the fleet
 * @return the ID of the created fleet
 */
suspend fun createFleet(catARN: String, fleetIdVal: String): String {
    val fleetRequest = CreateFleetRequest {
        fleetId = fleetIdVal
        signalCatalogArn = catARN
        description = "Built using the AWS For Kotlin"
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createFleet(fleetRequest)
        return response.id
    }
}

/**
 * Creates a signal catalog.
 *
 * @param signalCatalogName the name of the signal catalog to create the branch
 vehicle in
```

```
 * @return the ARN (Amazon Resource Name) of the created signal catalog
 */
suspend fun createbranchVehicle(signalCatalogName: String): String {
    delay(2000) // Wait for 2 seconds
    val branchVehicle = Branch {
        fullyQualifiedName = "Vehicle"
        description = "Root branch"
    }

    val branchPowertrain = Branch {
        fullyQualifiedName = "Vehicle.Powertrain"
        description = "Powertrain branch"
    }

    val sensorRPM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        description = "Engine RPM"
        dataType = NodeDataType.Double
        unit = "rpm"
    }

    val sensorKM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        description = "Vehicle Speed"
        dataType = NodeDataType.Double
        unit = "km/h"
    }

    // Wrap each specific node type (Branch and Sensor) into the sealed Node class
    // so they can be included in the CreateSignalCatalogRequest.
    val myNodes = listOf(
        Node.Branch(branchVehicle),
        Node.Branch(branchPowertrain),
        Node.Sensor(sensorRPM),
        Node.Sensor(sensorKM),
    )

    val request = CreateSignalCatalogRequest {
        name = signalCatalogName
        nodes = myNodes
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
```

```
        val response = fleetwiseClient.createSignalCatalog(request)
        return response.arn
    }
}

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 {
            println("Invalid input. Please try again.")
        }
    }
}
```

Aktionen

`createDecoderManifest`

Das folgende Codebeispiel zeigt die Verwendung`createDecoderManifest`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new decoder manifest.
 *
 * @param decName             the name of the decoder manifest
 * @param modelManifestArnVal the ARN of the model manifest
```

```
 * @return the ARN of the decoder manifest
 */
suspend fun createDecoderManifest(decName: String, modelManifestArnVal: String?):
 String {
    val interfaceIdVal = "can0"

    val canInter = CanInterface {
        name = "canInterface0"
        protocolName = "CAN"
        protocolVersion = "1.0"
    }

    val networkInterface = NetworkInterface {
        interfaceId = interfaceIdVal
        type = NetworkInterfaceType.CanInterface
        canInterface = canInter
    }

    val carRpmSig = CanSignal {
        messageId = 100
        isBigEndian = false
        isSigned = false
        startBit = 16
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val carSpeedSig = CanSignal {
        messageId = 101
        isBigEndian = false
        isSigned = false
        startBit = 0
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val engineRpmDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carRpmSig
    }
```

```
    val vehicleSpeedDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carSpeedSig
    }

    val request = CreateDecoderManifestRequest {
        name = decName
        modelManifestArn = modelManifestArnVal
        networkInterfaces = listOf(networkInterface)
        signalDecoders = listOf(engineRpmDecoder, vehicleSpeedDecoder)
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createDecoderManifest(request)
        return response.arn
    }
}
```

- API-Details finden Sie createDecoderManifestin der API-Referenz zum AWS SDK für Kotlin.

`createFleet`

Das folgende Codebeispiel zeigt die Verwendung`createFleet`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new fleet.
 *
 * @param catARN the Amazon Resource Name (ARN) of the signal catalog to associate
 with the fleet
```

```
 * @param fleetId the unique identifier for the fleet
 * @return the ID of the created fleet
 */
suspend fun createFleet(catARN: String, fleetIdVal: String): String {
    val fleetRequest = CreateFleetRequest {
        fleetId = fleetIdVal
        signalCatalogArn = catARN
        description = "Built using the AWS For Kotlin"
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createFleet(fleetRequest)
        return response.id
    }
}
```

- Weitere API-Informationen finden Sie unter createFleet in der API-Referenz zum AWS -SDK für Kotlin.

`createModelManifest`

Das folgende Codebeispiel zeigt, wie man es benutzt`createModelManifest`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a model manifest.
 *
 * @param name              the name of the model manifest to create
 * @param signalCatalogArn  the Amazon Resource Name (ARN) of the signal catalog
 * @param nodes             a list of nodes to include in the model manifest
 * @return a {@link CompletableFuture} that completes with the ARN of the created
 model manifest
```

```
 */
suspend fun createModelManifest(nameVal: String, signalCatalogArnVal: String,
 nodesList: List<Node>): String {
    val fqnList: List<String> = nodesList.map { node ->
        when (node) {
            is Node.Sensor -> node.asSensor().fullyQualifiedName
            is Node.Branch -> node.asBranch().fullyQualifiedName
            else -> throw RuntimeException("Unsupported node type")
        }
    }

    val request = CreateModelManifestRequest {
        name = nameVal
        signalCatalogArn = signalCatalogArnVal
        nodes = fqnList
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createModelManifest(request)
        return response.arn
    }
}
```

- API-Details finden Sie createModelManifestin der API-Referenz zum AWS SDK für Kotlin.

`createSignalCatalog`

Das folgende Codebeispiel zeigt die Verwendung`createSignalCatalog`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a signal catalog.
 *
```

```
 * @param signalCatalogName the name of the signal catalog to create the branch
 vehicle in
 * @return the ARN (Amazon Resource Name) of the created signal catalog
 */
suspend fun createbranchVehicle(signalCatalogName: String): String {
    delay(2000) // Wait for 2 seconds
    val branchVehicle = Branch {
        fullyQualifiedName = "Vehicle"
        description = "Root branch"
    }

    val branchPowertrain = Branch {
        fullyQualifiedName = "Vehicle.Powertrain"
        description = "Powertrain branch"
    }

    val sensorRPM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        description = "Engine RPM"
        dataType = NodeDataType.Double
        unit = "rpm"
    }

    val sensorKM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        description = "Vehicle Speed"
        dataType = NodeDataType.Double
        unit = "km/h"
    }

    // Wrap each specific node type (Branch and Sensor) into the sealed Node class
    // so they can be included in the CreateSignalCatalogRequest.
    val myNodes = listOf(
        Node.Branch(branchVehicle),
        Node.Branch(branchPowertrain),
        Node.Sensor(sensorRPM),
        Node.Sensor(sensorKM),
    )

    val request = CreateSignalCatalogRequest {
        name = signalCatalogName
        nodes = myNodes
    }
```

```
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.createSignalCatalog(request)
        return response.arn
    }
}
```

- API-Details finden Sie createSignalCatalogin der API-Referenz zum AWS SDK für Kotlin.

`createVehicle`

Das folgende Codebeispiel zeigt die Verwendung`createVehicle`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createVehicle(vecName: String, manifestArn: String?, decArn: String) {
    val request = CreateVehicleRequest {
        vehicleName = vecName
        modelManifestArn = manifestArn
        decoderManifestArn = decArn
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.createVehicle(request)
        println("Vehicle $vecName was created successfully.")
    }
}
```

- Weitere API-Informationen finden Sie unter createVehicle in der API-Referenz zum AWS -SDK für Kotlin.

`deleteDecoderManifest`

Das folgende Codebeispiel zeigt, wie man es benutzt`deleteDecoderManifest`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDecoderManifest(nameVal: String) {
    val request = DeleteDecoderManifestRequest {
        name = nameVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteDecoderManifest(request)
        println("$nameVal was successfully deleted")
    }
}
```

- API-Details finden Sie deleteDecoderManifestin der API-Referenz zum AWS SDK für Kotlin.

`deleteFleet`

Das folgende Codebeispiel zeigt die Verwendung`deleteFleet`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
 * Deletes a fleet based on the provided fleet ID.
 *
 * @param fleetId the ID of the fleet to be deleted
 */
suspend fun deleteFleet(fleetIdVal: String) {
    val request = DeleteFleetRequest {
        fleetId = fleetIdVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteFleet(request)
        println(" $fleetIdVal was successfully deleted")
    }
}
```

- Weitere API-Informationen finden Sie unter deleteFleet in der API-Referenz zum AWS -SDK für Kotlin.

`deleteModelManifest`

Das folgende Codebeispiel zeigt, wie man es benutzt`deleteModelManifest`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a model manifest.
 *
 * @param nameVal the name of the model manifest to delete
 */
suspend fun deleteModelManifest(nameVal: String) {
    val request = DeleteModelManifestRequest {
        name = nameVal
    }
```

```
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteModelManifest(request)
        println(" $nameVal was successfully deleted")
    }
}
```

- API-Details finden Sie deleteModelManifestin der API-Referenz zum AWS SDK für Kotlin.

`deleteSignalCatalog`

Das folgende Codebeispiel zeigt die Verwendung`deleteSignalCatalog`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a signal catalog.
 *
 * @param name the name of the signal catalog to delete
 */
suspend fun deleteSignalCatalog(catName: String) {
    val request = DeleteSignalCatalogRequest {
        name = catName
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteSignalCatalog(request)
        println(" $catName was successfully deleted")
    }
}
```

- API-Details finden Sie deleteSignalCatalogin der API-Referenz zum AWS SDK für Kotlin.

`deleteVehicle`

Das folgende Codebeispiel zeigt die Verwendung`deleteVehicle`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteVehicle(vecName: String) {
    val request = DeleteVehicleRequest {
        vehicleName = vecName
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.deleteVehicle(request)
        println("Vehicle $vecName was deleted successfully.")
    }
}
```

- Weitere API-Informationen finden Sie unter deleteVehicle in der API-Referenz zum AWS -SDK für Kotlin.

`getDecoderManifest`

Das folgende Codebeispiel zeigt, wie man es benutzt`getDecoderManifest`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Waits for the specified model manifest to become active.
 *
 * @param decNameVal the name of the model manifest to wait for
 */
suspend fun waitForDecoderManifestActive(decNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetDecoderManifestRequest {
                    name = decNameVal
                }

                val response = fleetwiseClient.getDecoderManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: INVALID #\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
```

```
        }
    }
}
```

- API-Details finden Sie getDecoderManifestin der API-Referenz zum AWS SDK für Kotlin.

`getModelManifest`

Das folgende Codebeispiel zeigt die Verwendung`getModelManifest`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Waits for the specified model manifest to become active.
 *
 * @param manifestName the name of the model manifest to wait for
 */
suspend fun waitForModelManifestActive(manifestNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetModelManifestRequest {
                    name = manifestNameVal
                }

                val response = fleetwiseClient.getModelManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft
```

```
                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: INVALID #
\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}
```

- API-Details finden Sie getModelManifestin der API-Referenz zum AWS SDK für Kotlin.

`getVehicle`

Das folgende Codebeispiel zeigt die Verwendung`getVehicle`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getVehicleDetails(vehicleNameVal: String) {
    val request = GetVehicleRequest {
```

```
        vehicleName = vehicleNameVal
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.getVehicle(request)
        val details = mapOf(
            "vehicleName" to response.vehicleName,
            "arn" to response.arn,
            "modelManifestArn" to response.modelManifestArn,
            "decoderManifestArn" to response.decoderManifestArn,
            "attributes" to response.attributes.toString(),
            "creationTime" to response.creationTime.toString(),
            "lastModificationTime" to response.lastModificationTime.toString(),
        )

        println("Vehicle Details:")
        for ((key, value) in details) {
            println("• %-20s : %s".format(key, value))
        }
    }
}
```

- Weitere API-Informationen finden Sie unter getVehicle in der API-Referenz zum AWS -SDK für Kotlin.

`listSignalCatalogNodes`

Das folgende Codebeispiel zeigt, wie man es benutzt`listSignalCatalogNodes`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Lists the signal catalog nodes asynchronously.
```

```
 *
 * @param signalCatalogName the name of the signal catalog
 * @return a CompletableFuture that, when completed, contains a list of nodes in the
 specified signal catalog
 * @throws CompletionException if an exception occurs during the asynchronous
 operation
 */
suspend fun listSignalCatalogNode(signalCatalogName: String): List<Node>? {
    val request = ListSignalCatalogNodesRequest {
        name = signalCatalogName
    }

    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogNodes(request)
        return response.nodes
    }
}
```

- API-Details finden Sie unter listSignalCatalogNodes in AWS SDK for Kotlin API-Referenz.

`updateDecoderManifest`

Das folgende Codebeispiel zeigt die Verwendung`updateDecoderManifest`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateDecoderManifest(nameVal: String) {
    val request = UpdateDecoderManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
```

```
        fleetwiseClient.updateDecoderManifest(request)
        println("$nameVal was successfully updated")
    }
}
```

- API-Details finden Sie updateDecoderManifestin der API-Referenz zum AWS SDK für Kotlin.

`updateModelManifest`

Das folgende Codebeispiel zeigt die Verwendung`updateModelManifest`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Updates the model manifest.
 *
 * @param nameVal the name of the model manifest to update
 */
suspend fun updateModelManifest(nameVal: String) {
    val request = UpdateModelManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient.fromEnvironment { region = "us-east-1" }.use
 { fleetwiseClient ->
        fleetwiseClient.updateModelManifest(request)
        println("$nameVal was successfully updated")
    }
}
```

- API-Details finden Sie updateModelManifestin der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon Keyspaces unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Keyspaces Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Keyspaces

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Keyspaces beginnen können.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- API-Details finden Sie ListKeyspacesin der API-Referenz zum AWS SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Keyspace und eine Tabelle. Das Tabellenschema enthält Filmdaten und die point-in-time Wiederherstellung ist aktiviert.
- Stellen Sie über eine sichere TLS-Verbindung mit SigV4-Authentifizierung eine Verbindung zum Keyspace her.
- Fragen Sie die Tabelle ab. Fügen Sie Filmdaten hinzu, rufen Sie sie ab und aktualisieren Sie sie.

- Aktualisieren Sie die Tabelle. Fügen Sie eine Spalte hinzu, um bereits gesehene Filme zu verfolgen.
- Stellen Sie den vorherigen Zustand der Tabelle wieder her und bereinigen Sie die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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 schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Aktionen

`CreateKeyspace`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateKeyspace`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- API-Details finden Sie CreateKeyspacein der API-Referenz zum AWS SDK für Kotlin.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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 schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- API-Details finden Sie CreateTablein der API-Referenz zum AWS SDK für Kotlin.

`DeleteKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyspace`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
```

```
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- API-Details finden Sie DeleteKeyspacein der API-Referenz zum AWS SDK für Kotlin.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- API-Details finden Sie DeleteTablein der API-Referenz zum AWS SDK für Kotlin.

`GetKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`GetKeyspace`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- API-Details finden Sie GetKeyspacein der API-Referenz zum AWS SDK für Kotlin.

`GetTable`

Das folgende Codebeispiel zeigt die Verwendung`GetTable`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
            }
        }
    }
}
```

- API-Details finden Sie GetTablein der API-Referenz zum AWS SDK für Kotlin.

`ListKeyspaces`

Das folgende Codebeispiel zeigt die Verwendung`ListKeyspaces`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- API-Details finden Sie ListKeyspacesin der API-Referenz zum AWS SDK für Kotlin.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }
```

```
    KeyspacesClient.fromEnvironment { 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}")
            }
    }
}
```

- API-Details finden Sie ListTablesin der API-Referenz zum AWS SDK für Kotlin.

`RestoreTable`

Das folgende Codebeispiel zeigt die Verwendung`RestoreTable`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- API-Details finden Sie RestoreTablein der API-Referenz zum AWS SDK für Kotlin.

`UpdateTable`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTable`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- API-Details finden Sie UpdateTablein der API-Referenz zum AWS SDK für Kotlin.

AWS KMS Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. AWS KMS

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAlias`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
```

```
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- API-Details finden Sie CreateAliasin der API-Referenz zum AWS SDK für Kotlin.

`CreateGrant`

Das folgende Codebeispiel zeigt die Verwendung`CreateGrant`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
```

```
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- API-Details finden Sie CreateGrantin der API-Referenz zum AWS SDK für Kotlin.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { 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
    }
}
```

- API-Details finden Sie CreateKeyin der API-Referenz zum AWS SDK für Kotlin.

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung`Decrypt`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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)
    }
}
```

- Weitere API-Informationen finden Sie unter Decrypt in der API-Referenz zum AWS -SDK für Kotlin.

`DescribeKey`

Das folgende Codebeispiel zeigt, wie man es benutzt`DescribeKey`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { 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}")
    }
}
```

- API-Details finden Sie DescribeKeyin der API-Referenz zum AWS SDK für Kotlin.

`DisableKey`

Das folgende Codebeispiel zeigt die Verwendung`DisableKey`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- API-Details finden Sie DisableKeyin der API-Referenz zum AWS SDK für Kotlin.

`EnableKey`

Das folgende Codebeispiel zeigt die Verwendung`EnableKey`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
```

```
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- API-Details finden Sie EnableKeyin der API-Referenz zum AWS SDK für Kotlin.

`Encrypt`

Das folgende Codebeispiel zeigt die Verwendung`Encrypt`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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)
    }
}
```

- Weitere API-Informationen finden Sie unter Encrypt in der API-Referenz zum AWS -SDK für Kotlin.

`ListAliases`

Das folgende Codebeispiel zeigt, wie man es benutzt`ListAliases`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }
```

```
    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- API-Details finden Sie ListAliasesin der API-Referenz zum AWS SDK für Kotlin.

`ListGrants`

Das folgende Codebeispiel zeigt die Verwendung`ListGrants`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- API-Details finden Sie ListGrantsin der API-Referenz zum AWS SDK für Kotlin.

`ListKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListKeys`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie ListKeysin der API-Referenz zum AWS SDK für Kotlin.

Lambda-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateFunction`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
    }
}
```

- API-Details finden Sie CreateFunctionin der API-Referenz zum AWS SDK für Kotlin.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- API-Details finden Sie DeleteFunctionin der API-Referenz zum AWS SDK für Kotlin.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}")
    }
}
```

- Weitere API-Informationen finden Sie unter Invoke in der API-Referenz zum AWS -SDK für Kotlin.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

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

- Amazon SNS

Beispiele für Amazon Location unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Location Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Location

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Location Service beginnen können.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
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 need to create a collection using the AWS Management
console. For information, see the following documentation.

https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:
            <colletionName>

        Where:
            colletionName - The Amazon location collection name.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }
    val colletionName = args[0]
    listGeofences(colletionName)
}

/**
 * Lists the geofences for the specified collection name.
 *
 * @param collectionName the name of the geofence collection
 */
suspend fun listGeofences(collectionName: String) {
    val request = ListGeofencesRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.listGeofences(request)
        val geofences = response.entries
        if (geofences.isNullOrEmpty()) {
```

```
            println("No Geofences found")
        } else {
            geofences.forEach { geofence ->
                println("Geofence ID: ${geofence.geofenceId}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - ListGeofenceCollections
  - ListGeofences

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon-Location-Karte.
- Erstellen Sie einen API-Schlüssel für Amazon Location.
- Zeigen Sie die Karten-URL an.
- Erstellen Sie eine Geofence-Sammlung.
- Speichern einer Geofence-Geometrie
- Erstellen einer Tracker-Ressource
- Aktualisieren der Position eines Geräts
- Rufen Sie die neueste Positionsaktualisierung für ein bestimmtes Gerät ab.
- Erstellen Sie einen Routenrechner.
- Ermitteln der Entfernung zwischen Seattle und Vancouver
- Verwenden Sie Amazon Location auf höherer Ebene APIs.
- Löschen Sie die Amazon-Location-Komponenten.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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 scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

        Where:
            mapName - The name of the map to create (e.g., "AWSMap").
            keyName - The name of the API key to create (e.g., "AWSApiKey").
            collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
            geoId - The geographic identifier used for the geofence or map (e.g.,
 "geoId").
            trackerName - The name of the tracker (e.g., "geoTracker").
            calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
            deviceId - The ID of the device (e.g., "iPhone-112356").
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(0)
    }
```

```
    val mapName = args[0]
    val keyName = args[1]
    val collectionName = args[2]
    val geoId = args[3]
    val trackerName = args[4]
    val calculatorName = args[5]
    val deviceId = args[6]

    println(
        """
    AWS Location Service is a fully managed service offered by Amazon Web Services
(AWS) that
    provides location-based services for developers. This service simplifies
    the integration of location-based features into applications, making it
    easier to build and deploy location-aware applications.

    The AWS Location Service offers a range of location-based services,
    including:

    - Maps: The service provides access to high-quality maps, satellite imagery,
      and geospatial data from various providers, allowing developers to
      easily embed maps into their applications.

    - Tracking: The Location Service enables real-time tracking of mobile devices,
      assets, or other entities, allowing developers to build applications
      that can monitor the location of people, vehicles, or other objects.

    - Geocoding: The service provides the ability to convert addresses or
      location names into geographic coordinates (latitude and longitude),
      and vice versa, enabling developers to integrate location-based search
      and routing functionality into their applications.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println(DASHES)
    println("1. Create an AWS Location Service map")
    println(
        """
        An AWS Location map can enhance the user experience of your
        application by providing accurate and personalized location-based
        features. For example, you could use the geocoding capabilities to
        allow users to search for and locate businesses, landmarks, or
        other points of interest within a specific region.
```

```
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    val mapArn = createMap(mapName)
    println("The Map ARN is: $mapArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    waitForInputToContinue(scanner)
    println("2. Create an AWS Location API key")
    println(
        """
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.

        """.trimIndent(),
    )
    val keyArn = createKey(keyName, mapArn)
    println("The Key ARN is: $keyArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("3. Display Map URL")
    println(
        """
        In order to get the MAP URL, you need to get the API Key value.
        You can get the key value using the AWS Management Console under
        Location Services. This operation cannot be completed using the
        AWS SDK. For more information about getting the key value, see
        the AWS Location Documentation.
        """.trimIndent(),
    )
    val mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/$mapName/tiles/{z}/
{x}/{y}?key={KeyValue}"
    println("Embed this URL in your Web app: $mapUrl")
    println("")
    waitForInputToContinue(scanner)
    println(DASHES)
```

```
    println(DASHES)
    println("4. Create a geofence collection, which manages and stores geofences.")
    waitForInputToContinue(scanner)
    val collectionArn: String =
        createGeofenceCollection(collectionName)
    println("The geofence collection was successfully created: $collectionArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println("5. Store a geofence geometry in a given geofence collection.")
    println(
        """
        An AWS Location geofence is a virtual boundary that defines a geographic
area
        on a map. It is a useful feature for tracking the location of
        assets or monitoring the movement of objects within a specific region.

        To define a geofence, you need to specify the coordinates of a
        polygon that represents the area of interest. The polygon must be
        defined in a counter-clockwise direction, meaning that the points of
        the polygon must be listed in a counter-clockwise order.

        This is a requirement for the AWS Location service to correctly
        interpret the geofence and ensure that the location data is
        accurately processed within the defined area.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    putGeofence(collectionName, geoId)
    println("Successfully created geofence: $geoId")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Create a tracker resource which lets you retrieve current and
historical location of devices.")
    waitForInputToContinue(scanner)
    val trackerArn: String = createTracker(trackerName)
    println("Successfully created tracker. ARN: $trackerArn")
    waitForInputToContinue(scanner)
    println(DASHES)
```

```
    println(DASHES)
    println("7. Update the position of a device in the location tracking system.")
    println(
        """
        The AWS location service does not enforce a strict format for deviceId, but
it must:
            - Be a string (case-sensitive).
            - Be 1–100 characters long.
            - Contain only:
            - Alphanumeric characters (A-Z, a-z, 0-9)
            - Underscores (_)
            - Hyphens (-)
            - Be the same ID used when sending and retrieving positions.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    updateDevicePosition(trackerName, deviceId)
    println("$deviceId was successfully updated in the location tracking system.")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("8. Retrieve the most recent position update for a specified device.")
    waitForInputToContinue(scanner)
    val response = getDevicePosition(trackerName, deviceId)
    println("Successfully fetched device position: ${response.position}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("9. Create a route calculator.")
    waitForInputToContinue(scanner)
    val routeResponse = createRouteCalculator(calculatorName)
    println("Route calculator created successfully: ${routeResponse.calculatorArn}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("10. Determine the distance in kilometers between Seattle and Vancouver
using the route calculator.")
    waitForInputToContinue(scanner)
    val responseDis = calcDistance(calculatorName)
```

```
    println("Successfully calculated route. The distance in kilometers is
${responseDis.summary?.distance}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("11. Use the GeoPlacesClient to perform additional operations.")
    println(
        """
        This scenario will show use of the GeoPlacesClient that enables
        location search and geocoding capabilities for your applications.

        We are going to use this client to perform these AWS Location tasks:
            - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
            - Place Search (searchText): Finds places based on search queries.
            - Nearby Search (searchNearby): Finds places near a specific location.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println("First we will perform a Reverse Geocoding operation")
    waitForInputToContinue(scanner)
    reverseGeocode()

    println("Now we are going to perform a text search using coffee shop.")
    waitForInputToContinue(scanner)
    searchText("coffee shop")
    waitForInputToContinue(scanner)

    println("Now we are going to perform a nearby Search.")
    waitForInputToContinue(scanner)
    searchNearby()
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS Location Services resources.")
    println("Would you like to delete the AWS Location Services resources? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        deleteMap(mapName)
        deleteKey(keyName)
```

```
        deleteGeofenceCollection(collectionName)
        deleteTracker(trackerName)
        deleteRouteCalculator(calculatorName)
    } else {
        println("The AWS resources will not be deleted.")
    }
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println(" This concludes the AWS Location Service scenario.")
    println(DASHES)
}

/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}


/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}
```

```
/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
```

```
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}


/**
 * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
 * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
 * The results are processed and printed once the search completes successfully.
 */
suspend fun searchNearby() {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    // Set up the request for searching nearby places.
    val request = SearchNearbyRequest {
        this.queryPosition = queryPosition
        this.queryRadius = 1000L
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchNearby(request)

        // Process the response and print the results.
        response.resultItems?.forEach { result ->
            println("Title: ${result.title}")
            println("Address: ${result.address?.label}")
            println("Distance: ${result.distance} meters")
            println("-------------------------")
        }
    }
}


/**
 * Searches for a place using the provided search query and prints the detailed
 information of the first result.
```

```
 *
 * @param searchQuery the search query to be used for the place search (ex, coffee
 shop)
 */
suspend fun searchText(searchQuery: String) {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    val request = SearchTextRequest {
        this.queryText = searchQuery
        this.biasPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchText(request)

        response.resultItems?.firstOrNull()?.let { result ->
            val placeId = result.placeId // Get Place ID
            println("Found Place with id: $placeId")

            // Fetch detailed info using getPlace.
            val getPlaceRequest = GetPlaceRequest {
                this.placeId = placeId
            }

            val placeResponse = client.getPlace(getPlaceRequest)

            // Print detailed place information.
            println("Detailed Place Information:")
            println("Title: ${placeResponse.title}")
            println("Address: ${placeResponse.address?.label}")

            // Print each food type (if any).
            placeResponse.foodTypes?.takeIf { it.isNotEmpty() }?.let {
                println("Food Types:")
                it.forEach { foodType ->
                    println("  - $foodType")
                }
            } ?: run {
                println("No food types available.")
            }

            println("-------------------------")
```

```
        }
    }
}

/**
 * Performs reverse geocoding using the AWS Geo Places API.
 * Reverse geocoding is the process of converting geographic coordinates (latitude
 and longitude) to a human-readable address.
 * This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.
 */
suspend fun reverseGeocode() {
    val latitude = 37.7749
    val longitude = -122.4194
    println("Use latitude 37.7749 and longitude -122.4194")

    // AWS expects [longitude, latitude].
    val queryPosition = listOf(longitude, latitude)
    val request = ReverseGeocodeRequest {
        this.queryPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.reverseGeocode(request)
        response.resultItems?.forEach { result ->
            println("The address is: ${result.address?.label}")
        }
    }
}


/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
```

```
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}

/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}

/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}

/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}

/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
```

```
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}

/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}

/**
 * Creates a new geofence collection.
 *
```

```
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}

/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}

/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
```

```
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}

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 {
            println("Invalid input. Please try again.")
        }
    }
}
```

Aktionen

`BatchUpdateDevicePosition`

Das folgende Codebeispiel zeigt, wie man es benutzt`BatchUpdateDevicePosition`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

- API-Details finden Sie BatchUpdateDevicePositionin der API-Referenz zum AWS SDK für Kotlin.

`CalculateRoute`

Das folgende Codebeispiel zeigt die Verwendung`CalculateRoute`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}
```

- API-Details finden Sie CalculateRoutein der API-Referenz zum AWS SDK für Kotlin.

`CreateGeofenceCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateGeofenceCollection`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}
```

- API-Details finden Sie CreateGeofenceCollectionin der API-Referenz zum AWS SDK für Kotlin.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}
```

- API-Details finden Sie CreateKeyin der API-Referenz zum AWS SDK für Kotlin.

`CreateMap`

Das folgende Codebeispiel zeigt die Verwendung`CreateMap`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}
```

- API-Details finden Sie CreateMapin der API-Referenz zum AWS SDK für Kotlin.

`CreateRouteCalculator`

Das folgende Codebeispiel zeigt die Verwendung`CreateRouteCalculator`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}
```

- API-Details finden Sie CreateRouteCalculatorin der API-Referenz zum AWS SDK für Kotlin.

`CreateTracker`

Das folgende Codebeispiel zeigt die Verwendung`CreateTracker`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}
```

- API-Details finden Sie CreateTrackerin der API-Referenz zum AWS SDK für Kotlin.

`DeleteGeofenceCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGeofenceCollection`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a geofence collection.
 *
```

```
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

- API-Details finden Sie DeleteGeofenceCollectionin der API-Referenz zum AWS SDK für Kotlin.

`DeleteKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKey`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
```

```
        println("The key $keyName was deleted.")
    }
}
```

- API-Details finden Sie DeleteKeyin der API-Referenz zum AWS SDK für Kotlin.

`DeleteMap`

Das folgende Codebeispiel zeigt die VerwendungDeleteMap.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

- API-Details finden Sie DeleteMapin der API-Referenz zum AWS SDK für Kotlin.

`DeleteRouteCalculator`

Das folgende Codebeispiel zeigt die VerwendungDeleteRouteCalculator.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}
```

- API-Details finden Sie DeleteRouteCalculatorin der API-Referenz zum AWS SDK für Kotlin.

`DeleteTracker`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTracker`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}
```

- API-Details finden Sie DeleteTrackerin der API-Referenz zum AWS SDK für Kotlin.

`GetDevicePosition`

Das folgende Codebeispiel zeigt die VerwendungGetDevicePosition.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}
```

- API-Details finden Sie GetDevicePositionin der API-Referenz zum AWS SDK für Kotlin.

`PutGeofence`

Das folgende Codebeispiel zeigt die VerwendungPutGeofence.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
```

```
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}
```

- API-Details finden Sie PutGeofencein der API-Referenz zum AWS SDK für Kotlin.

MediaConvert Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. MediaConvert

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInput1: String,
): String? {
    // Step 1: Describe endpoints to get the MediaConvert endpoint URL
    val describeResponse = mcClient.describeEndpoints(
        DescribeEndpointsRequest {
            maxResults = 1
        },
    )

    val endpointUrl = describeResponse.endpoints?.firstOrNull()?.url
        ?: error("No MediaConvert endpoint found")

    // Step 2: Create MediaConvert client with resolved endpoint
    val mediaConvert = MediaConvertClient.fromEnvironment {
        region = "us-west-2"
        endpointProvider = MediaConvertEndpointProvider {
            Endpoint(endpointUrl)
        }
    }

    // Output destination folder in S3 - put in 'output/' folder beside input
    val outputDestination = fileInput1.substringBeforeLast('/') + "/output/"

    // Step 3: Create the job request with minimal valid video codec settings
    val jobRequest = CreateJobRequest {
        role = mcRoleARN
        settings = JobSettings {
            inputs = listOf(
                Input {
                    fileInput = fileInput1
                },
            )
            outputGroups = listOf(
                OutputGroup {
                    outputGroupSettings = OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings = FileGroupSettings {
                            destination = outputDestination
                        }
                    }
```

```
                    outputs = listOf(
                        Output {
                            containerSettings = ContainerSettings {
                                container = ContainerType.Mp4
                            }
                            videoDescription = VideoDescription {
                                width = 1280
                                height = 720
                                codecSettings = VideoCodecSettings {
                                    codec = VideoCodec.H264
                                    h264Settings = H264Settings {
                                        rateControlMode = H264RateControlMode.Qvbr
                                        qvbrSettings = H264QvbrSettings {
                                            qvbrQualityLevel = 7
                                        }
                                        maxBitrate = 5_000_000
                                        codecLevel = H264CodecLevel.Auto
                                        codecProfile = H264CodecProfile.Main
                                        framerateControl =
 H264FramerateControl.InitializeFromSource
                                    }
                                }
                            }
                        },
                    )
                },
            )
        }
    }

    // Step 4: Call MediaConvert to create the job
    val response = mediaConvert.createJob(jobRequest)

    // Return the job ID or null if not found
    return response.job?.id
}
```

- API-Details finden Sie CreateJobin der API-Referenz zum AWS SDK für Kotlin.

`GetJob`

Das folgende Codebeispiel zeigt die Verwendung`GetJob`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificJob(mcClient: MediaConvertClient, jobId: String) {
    // 1. Discover the correct endpoint
    val res = mcClient.describeEndpoints(DescribeEndpointsRequest { maxResults =
 1 })
    var endpointUrl = res.endpoints?.firstOrNull()?.url
        ?: error(" No MediaConvert endpoint found")

    // 2. Create a new client using the endpoint
    val clientWithEndpoint = MediaConvertClient {
        region = "us-west-2"
        endpointUrl = endpointUrl
    }

    // 3. Get the job details
    val jobResponse = clientWithEndpoint.getJob(GetJobRequest { id = jobId })
    val job = jobResponse.job

    println("Job status: ${job?.status}")
    println("Job ARN: ${job?.arn}")
    println("Output group count: ${job?.settings?.outputGroups?.size}")
}
```

- API-Details finden Sie GetJobin der API-Referenz zum AWS SDK für Kotlin.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}")
        }
    }
```

```
}
```

- API-Details finden Sie ListJobsin der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon Pinpoint unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Pinpoint Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateApp`

Das folgende Codebeispiel zeigt die Verwendung. `CreateApp`

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
```

```
            name = applicationName
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- API-Details finden Sie CreateAppin der API-Referenz zum AWS SDK für Kotlin.

`CreateCampaign`

Das folgende Codebeispiel zeigt die Verwendung`CreateCampaign`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val scheduleOb =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessageOb =
        Message {
            action = Action.OpenApp
            body = "My message body"
```

```
            title = "My message title"
        }

    val messageConfigurationOb =
        MessageConfiguration {
            defaultMessage = defaultMessageOb
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = scheduleOb
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfigurationOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- API-Details finden Sie CreateCampaignin der API-Referenz zum AWS SDK für Kotlin.

`CreateSegment`

Das folgende Codebeispiel zeigt die Verwendung`CreateSegment`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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 writeSegmentRequestOb =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensionsOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
```

```
                    writeSegmentRequest = writeSegmentRequestOb
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- API-Details finden Sie CreateSegmentin der API-Referenz zum AWS SDK für Kotlin.

`DeleteApp`

Das folgende Codebeispiel zeigt die Verwendung`DeleteApp`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- API-Details finden Sie DeleteAppin der API-Referenz zum AWS SDK für Kotlin.

`DeleteEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEndpoint`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- API-Details finden Sie DeleteEndpointin der API-Referenz zum AWS SDK für Kotlin.

`GetEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`GetEndpoint`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient.fromEnvironment { 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)
    }
}
```

- API-Details finden Sie GetEndpointin der API-Referenz zum AWS SDK für Kotlin.

`GetSegments`

Das folgende Codebeispiel zeigt die Verwendung`GetSegments`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSegs(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

- API-Details finden Sie GetSegmentsin der API-Referenz zum AWS SDK für Kotlin.

`SendMessages`

Das folgende Codebeispiel zeigt die Verwendung`SendMessages`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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 = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient.fromEnvironment { region = "us-east-1" }.use { pinpointemail
 ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- API-Details finden Sie SendMessagesin der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon RDS unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon RDS Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-west-2" }.use
 { secretsClient ->
```

```
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateDBInstance
  - DBParameterGruppe erstellen
  - CreateDBSnapshot
  - LöschenDBInstance
  - DBParameterGruppe löschen
  - DBEngineVersionen beschreiben
  - Beschreiben DBInstances
  - Beschreiben Sie DBParameter Gruppen
  - Beschreiben DBParameters
  - Beschreiben DBSnapshots
  - DescribeOrderableDBInstanceOptionen
  - DBParameterGruppe ändern

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDatabaseInstance(
```

```
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            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.fromEnvironment { 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.fromEnvironment { 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!")
    }
}
```

- API-Details finden Sie unter API-Referenz DBInstance im AWS SDK für Kotlin erstellen.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
```

```
}
```

- API-Details finden Sie unter Löschen DBInstance im AWS SDK für die Kotlin-API-Referenz.

`DescribeAccountAttributes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAccountAttributes`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAccountAttributes() {
    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DescribeAccountAttributesin der API-Referenz zum AWS SDK für Kotlin.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeInstances() {
    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- Einzelheiten zur API finden Sie unter DBInstancesDescribe in AWS SDK for Kotlin API-Referenz.

`ModifyDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBInstance`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
```

```
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- API-Details finden Sie unter Modify DBInstance in AWS SDK for Kotlin API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

> Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.
>
> Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.
>
> In diesem Beispiel verwendete Dienste
>
> - Aurora
> - Amazon RDS
> - Amazon RDS Data Service
> - Amazon SES

Beispiele für Amazon Redshift unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Redshift Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung. `CreateCluster`

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den -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.fromEnvironment { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- API-Details finden Sie CreateClusterin der API-Referenz zum AWS SDK für Kotlin.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
```

```
            skipFinalClusterSnapshot = true
        }

    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- API-Details finden Sie DeleteClusterin der API-Referenz zum AWS SDK für Kotlin.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Beschreiben Sie den Cluster.

```
suspend fun describeRedshiftClusters() {
    RedshiftClient.fromEnvironment { 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}")
            }
        }
    }
}
```

- API-Details finden Sie DescribeClustersin der API-Referenz zum AWS SDK für Kotlin.

`ModifyCluster`

Das folgende Codebeispiel zeigt die Verwendung`ModifyCluster`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ändern Sie einen 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",
        )
    }
}
```

- API-Details finden Sie ModifyClusterin der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Erstellen einer Webanwendung zur Verfolgung von Amazon-Redshift-Daten

Das folgende Codebeispiel zeigt, wie Sie eine Webanwendung erstellen, die Arbeitselemente mit einer Amazon-Redshift-Datenbank verfolgt und darüber berichtet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Redshift Redshift-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Beispiele für Amazon Rekognition unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CompareFaces`

Das folgende Codebeispiel zeigt die Verwendung. `CompareFaces`

Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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 uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- API-Details finden Sie CompareFacesin der API-Referenz zum AWS SDK für Kotlin.

`CreateCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateCollection`.

Weitere Informationen finden Sie unter Erstellen einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
```

```
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- API-Details finden Sie CreateCollectionin der API-Referenz zum AWS SDK für Kotlin.

`DeleteCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCollection`.

Weitere Informationen finden Sie unter Löschen einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- API-Details finden Sie DeleteCollectionin der API-Referenz zum AWS SDK für Kotlin.

`DeleteFaces`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFaces`.

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- API-Details finden Sie DeleteFacesin der API-Referenz zum AWS SDK für Kotlin.

`DescribeCollection`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCollection`.

Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { 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}")
    }
}
```

- API-Details finden Sie DescribeCollectionin der API-Referenz zum AWS SDK für Kotlin.

`DetectFaces`

Das folgende Codebeispiel zeigt die Verwendung`DetectFaces`.

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
```

```
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DetectFacesin der API-Referenz zum AWS SDK für Kotlin.

`DetectLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectLabels`.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
```

```
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- API-Details finden Sie DetectLabelsin der API-Referenz zum AWS SDK für Kotlin.

`DetectModerationLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectModerationLabels`.

Weitere Informationen finden Sie unter Erkennen von unangemessenen Bildern.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
```

```
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DetectModerationLabelsin der API-Referenz zum AWS SDK für Kotlin.

`DetectText`

Das folgende Codebeispiel zeigt die Verwendung`DetectText`.

Weitere Informationen finden Sie unter Erkennen von Text in einem Bild.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie DetectTextin der API-Referenz zum AWS SDK für Kotlin.

`IndexFaces`

Das folgende Codebeispiel zeigt die Verwendung`IndexFaces`.

Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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")
            }
        }
    }
}
```

- API-Details finden Sie IndexFacesin der API-Referenz zum AWS SDK für Kotlin.

`ListCollections`

Das folgende Codebeispiel zeigt die Verwendung`ListCollections`.

Weitere Informationen finden Sie unter Sammlungen auflisten.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- API-Details finden Sie ListCollectionsin der API-Referenz zum AWS SDK für Kotlin.

`ListFaces`

Das folgende Codebeispiel zeigt die Verwendung`ListFaces`.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- API-Details finden Sie ListFacesin der API-Referenz zum AWS SDK für Kotlin.

`RecognizeCelebrities`

Das folgende Codebeispiel zeigt die Verwendung`RecognizeCelebrities`.

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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.")
    }
}
```

- API-Details finden Sie RecognizeCelebritiesin der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

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

Informationen in Videos erkennen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie Amazon-Rekognition-Aufträge, um Elemente wie Personen, Objekte und Text in Videos zu erkennen.
- Überprüfen Sie den Auftragsstatus, bis die Aufträge abgeschlossen sind.
- Gibt die Liste der von jedem Auftrag erkannten Elemente aus.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erkennen von Gesichtern in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient.fromEnvironment { 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.fromEnvironment { 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}")
        }
```

```
    }
}
```

Erkennen von unangemessenen oder anstößigen Inhalten in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient.fromEnvironment { 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)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection

- StartCelebrityRecognition
- StartContentModeration
- StartLabelDetection
- StartPersonTracking
- StartSegmentDetection
- StartTextDetection

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Beispiele für die Route-53-Domainregistrierung unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit der Route 53-Domänenregistrierung verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Route-53-Domainregistrierung

Das folgende Codebeispiel zeigt, wie Sie mit der Route 53-Domänenregistrierung beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 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.fromEnvironment { 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}")
            }
    }
}
```

- API-Details finden Sie ListPricesin der API-Referenz zum AWS SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Auflisten der aktuellen Domains und der Vorgänge des letzten Jahres
- Anzeigen der Abrechnung für das vergangene Jahr und der Preise für Domaintypen
- Abrufen von Domainvorschlägen
- Überprüfen der Verfügbarkeit und Übertragbarkeit von Domains
- Optional: Anfordern einer Domainregistrierung
- Abrufen eines Vorgangsdetails
- Optional: Abrufen eines Domaindetails

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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 xxxxxxx not found in xxxxxxx account.")
    getDomainDetails(domainSuggestion)
    println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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}")
            }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail

- GetDomainSuggestions
- GetOperationDetail
- ListDomains
- ListOperations
- ListPrices
- RegisterDomain
- ViewBilling

Aktionen

`CheckDomainAvailability`

Das folgende Codebeispiel zeigt, wie man es benutzt`CheckDomainAvailability`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- API-Details finden Sie CheckDomainAvailabilityin der API-Referenz zum AWS SDK für Kotlin.

`CheckDomainTransferability`

Das folgende Codebeispiel zeigt die Verwendung`CheckDomainTransferability`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- Einzelheiten zur API finden Sie CheckDomainTransferabilityin der API-Referenz zum AWS SDK für Kotlin.

`GetDomainDetail`

Das folgende Codebeispiel zeigt die Verwendung`GetDomainDetail`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { 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}")
    }
}
```

- Einzelheiten zur API finden Sie GetDomainDetailin der API-Referenz zum AWS SDK für Kotlin.

`GetDomainSuggestions`

Das folgende Codebeispiel zeigt die Verwendung`GetDomainSuggestions`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { 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(" ")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetDomainSuggestionsin der API-Referenz zum AWS SDK für Kotlin.

`GetOperationDetail`

Das folgende Codebeispiel zeigt die Verwendung`GetOperationDetail`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- Einzelheiten zur API finden Sie GetOperationDetailin der API-Referenz zum AWS SDK für Kotlin.

`ListDomains`

Das folgende Codebeispiel zeigt die Verwendung`ListDomains`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { 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}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListDomainsin der API-Referenz zum AWS SDK für Kotlin.

`ListOperations`

Das folgende Codebeispiel zeigt die Verwendung`ListOperations`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListOperationsin der API-Referenz zum AWS SDK für Kotlin.

`ListPrices`

Das folgende Codebeispiel zeigt die Verwendung`ListPrices`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllPrices(domainType: String?) {
```

```
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { 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}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListPricesin der API-Referenz zum AWS SDK für Kotlin.

`RegisterDomain`

Das folgende Codebeispiel zeigt die Verwendung`RegisterDomain`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- Einzelheiten zur API finden Sie RegisterDomainin der API-Referenz zum AWS SDK für Kotlin.

`ViewBilling`

Das folgende Codebeispiel zeigt die Verwendung`ViewBilling`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ViewBillingin der API-Referenz zum AWS SDK für Kotlin.

Amazon-S3-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
```

```
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt, wie man es benutzt`CopyObject`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun copyBucketObject(
    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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- Einzelheiten zur API finden Sie CopyObjectin der API-Referenz zum AWS SDK für Kotlin.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
```

```
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- Einzelheiten zur API finden Sie CreateBucketin der API-Referenz zum AWS SDK für Kotlin.

`CreateMultiRegionAccessPoint`

Das folgende Codebeispiel zeigt die Verwendung`CreateMultiRegionAccessPoint`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie den S3-Steuerungs-Client so, dass er eine Anfrage an die Region us-west-2 sendet.

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

Erstellen Sie den Multi-Region-Zugangspunkt.

```
    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"
    }
```

Warten Sie, bis der Multi-Region-Zugangspunkt verfügbar ist.

```
        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)
            }
        }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.
- Einzelheiten zur API finden Sie CreateMultiRegionAccessPointin der API-Referenz zum AWS SDK für Kotlin.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der API-Referenz zum AWS SDK für Kotlin.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der API-Referenz zum AWS SDK für Kotlin.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
```

```
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der API-Referenz zum AWS SDK für Kotlin.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
```

```
}
```

- Einzelheiten zur API finden Sie GetObjectin der API-Referenz zum AWS SDK für Kotlin.

`GetObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectAcl`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetObjectAclin der API-Referenz zum AWS SDK für Kotlin.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { 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
```

- API-Details finden Sie unter ListObjectsV2 im AWS SDK für die Kotlin-API-Referenz.

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAcl`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- Einzelheiten zur API finden Sie PutBucketAclin der API-Referenz zum AWS SDK für Kotlin.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- Einzelheiten zur API finden Sie PutObjectin der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine vorsignierte `GetObject`-Anfrage und verwenden Sie die URL, um ein Objekt herunterzuladen.

```
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
}
```

Erstellen Sie eine `GetObject`-vorsignierte Anfrage mit erweiterten Optionen.

```
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
}
```

Erstellen Sie eine vorsignierte `PutObject`-Anfrage und verwenden Sie sie, um ein Objekt hochzuladen.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)
```

```
    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Abrufen eines Objekts von einem Multi-Region-Zugangspunkt

Das folgende Codebeispiel zeigt, wie Sie ein Objekt von einem Multi-Region-Zugangspunkt abrufen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie den S3-Client so, dass er den Signaturalgorithmus Asymmetric Sigv4 (SigV4a) verwendet.

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a) signing
algorithm.
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

Verwenden Sie den ARN für den Multi-Region-Zugangspunkt eines Bucket-Namens.

```
    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
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.
- Einzelheiten zur API finden Sie GetObjectin der API-Referenz zum AWS SDK für Kotlin.

SageMaker KI-Beispiele, die SDK für Kotlin verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit SageMaker KI Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo AI SageMaker

Das folgende Codebeispiel zeigt, wie Sie mit SageMaker KI beginnen können.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listBooks() {
    SageMakerClient.fromEnvironment { region = "us-west-2" }.use { sageMakerClient -
>
```

```
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListNotebookInstancesin der API-Referenz zum AWS SDK für Kotlin.

Aktionen

`CreatePipeline`

Das folgende Codebeispiel zeigt die Verwendung`CreatePipeline`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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)
        }
    }
}
```

- Einzelheiten zur API finden Sie CreatePipelinein der API-Referenz zum AWS SDK für Kotlin.

`DeletePipeline`

Das folgende Codebeispiel zeigt die Verwendung`DeletePipeline`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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")
```

```
    }
}
```

- Einzelheiten zur API finden Sie DeletePipelinein der API-Referenz zum AWS SDK für Kotlin.

`DescribePipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribePipelineExecution`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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")
}
```

- Einzelheiten zur API finden Sie DescribePipelineExecutionin der API-Referenz zum AWS SDK für Kotlin.

`StartPipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartPipelineExecution`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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
    }
}
```

- Einzelheiten zur API finden Sie StartPipelineExecutionin der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Erste Schritte mit Geodatenaufträgen und Pipelines

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Ressourcen für eine Pipeline ein.
- Richten Sie eine Pipeline ein, die einen Geodatenauftrag ausführt.
- Starten Sie eine Pipeline-Ausführung.
- Überwachen Sie den Status der Ausführung.
- Zeigen Sie die Ausgabe der Pipeline an.
- Bereinigen Sie die Ressourcen.

Weitere Informationen finden Sie unter SageMaker Pipelines mithilfe AWS SDKs von Community.aws erstellen und ausführen.

SDK für Kotlin

> **Note**
>
> Weitere Informationen finden Sie unter. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
```

```
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
          <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
 <queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
 <pipelineName>

    Where:
        sageMakerRoleName - The name of the Amazon SageMaker role.
        lambdaRoleName - The name of the AWS Lambda role.
        functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).
        functionKey - The name of the Amazon S3 key name that represents the Lambda
 function (for example, SageMakerLambda.zip).
        queueName - The name of the Amazon Simple Queue Service (Amazon SQS) queue.
        bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
 bucket.
        bucketFunction - The name of the Amazon S3 bucket that contains the Lambda
 ZIP file.
        lnglatData - The file location of the latlongtest.csv file required for this
 use case.
        spatialPipelinePath - The file location of the GeoSpatialPipeline.json file
 required for this use case.
        pipelineName - The name of the pipeline to create (for example, sagemaker-
sdk-example-pipeline).
    """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]
```

```
    val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

    println(DASHES)
    println("Welcome to the Amazon SageMaker pipeline example scenario.")
    println(
        """
         This example workflow will guide you through setting up and running an
         Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and an
         Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
         reverse geocode addresses in an input file and store the results in an
export file.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("First, we will set up the roles, functions, and queue needed by the
SageMaker pipeline.")
    val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
    val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
    val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
    val queueUrl = checkQueue(queueName, functionName)
    println(DASHES)

    println(DASHES)
    println("Setting up bucket $bucketName")
    if (!checkBucket(bucketName)) {
        setupBucket(bucketName)
        println("Put $lnglatData into $bucketName")
        val objectKey = "samplefiles/latlongtest.csv"
        putS3Object(bucketName, objectKey, lnglatData)
    }
    println(DASHES)

    println(DASHES)
    println("Now we can create and run our pipeline.")
    setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn, pipelineName)
    val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
    println("The pipeline execution ARN value is $pipelineExecutionARN")
    waitForPipelineExecution(pipelineExecutionARN)
    println("Wait 30 secs to get output results $bucketName")
    TimeUnit.SECONDS.sleep(30)
```

```
    getOutputResults(bucketName)
    println(DASHES)

    println(DASHES)
    println(
        """
             The pipeline has completed. To view the pipeline and runs in SageMaker
 Studio, follow these instructions:
             https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-studio.html
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("Do you want to delete the AWS resources used in this Workflow? (y/n)")
    val `in` = Scanner(System.`in`)
    val delResources = `in`.nextLine()
    if (delResources.compareTo("y") == 0) {
        println("Lets clean up the AWS resources. Wait 30 seconds")
        TimeUnit.SECONDS.sleep(30)
        deleteEventSourceMapping(functionName)
        deleteSQSQueue(queueName)
        listBucketObjects(bucketName)
        deleteBucket(bucketName)
        delLambdaFunction(functionName)
        deleteLambdaRole(lambdaRoleName)
        deleteSagemakerRole(sageMakerRoleName)
        deletePipeline(pipelineName)
    } else {
        println("The AWS Resources were not deleted!")
    }
    println(DASHES)

    println(DASHES)
    println("SageMaker pipeline scenario is complete.")
    println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }
```

```
    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}

suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
```

```
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun delLambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}
```

```
suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
        bucket = bucketNameVal
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
```

```
                    eventSourceMapping = event.uuid.toString()
                }
            }
        }
    }

    val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
        uuid = eventSourceMapping
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
        println("The event mapping is deleted!")
    }
}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3Objects: List<Object>? = response.contents
        if (s3Objects != null) {
            for (`object` in s3Objects) {
                if (bucketName != null) {
```

```
                    readObject(bucketName, (`object`.key))
                }
            }
        }
    }
}

suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
```

```
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
```

```
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
```

```
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            println("$bucketName exists")
```

```
            return true
        }
    } catch (e: S3Exception) {
        println("Bucket does not exist")
    }
    return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }
    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
 lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
 successful")
    }
}
```

```
// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
    val queueAtt: MutableMap<String, String> = HashMap()
    queueAtt.put("DelaySeconds", "5")
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
 { queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a new
 queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
 created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
```

```
        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
}

suspend fun createNewFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
 function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
```

```
            val response = lambdaClient.getFunction(functionRequest)
            functionArn = response.configuration?.functionArn.toString()
            println("$functionArn exists")
        }
    } catch (e: LambdaException) {
        println(e.message + " A new function will be created")
        functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
 myHandler, myRole)
    }
    return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
```

```
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in sageMakerRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        System.out.println("Role ready with ARN ${roleResult.role?.arn}")
        return roleResult.role?.arn.toString()
    }
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
```

```
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createLambdaRole(roleNameVal)
    }

    return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
```

```
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        println("Role ready with ARN " + roleResult.role?.arn)
        return roleResult.role?.arn.toString()
    }
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<String>(5)
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy"
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
    return lambdaRolePolicies
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Beispiele für Secrets Manager unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit Secrets Manager verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`GetSecretValue`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use
 { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
```

```
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

- Einzelheiten zur API finden Sie GetSecretValuein der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon SES unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon SES Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB

- Amazon SES

Erstellen einer Webanwendung zur Verfolgung von Amazon-Redshift-Daten

Das folgende Codebeispiel zeigt, wie Sie eine Webanwendung erstellen, die Arbeitselemente mit einer Amazon-Redshift-Datenbank verfolgt und darüber berichtet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Redshift Redshift-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service

- Amazon SES

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Beispiele für Amazon SNS unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon SNS.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die VerwendungCreateTopic.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Kotlin.

`DeleteTopic`

Das folgende Codebeispiel zeigt die VerwendungDeleteTopic.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Kotlin.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
```

```
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSNSSubscriptions() {
    SnsClient.fromEnvironment { 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}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der API-Referenz zum AWS SDK für Kotlin.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSNSTopics() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

`Publish`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
```

```
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für Kotlin.

`SetTopicAttributes`

Das folgende Codebeispiel zeigt, wie man es benutzt`SetTopicAttributes`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
```

```
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

`Subscribe`

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
suspend fun subLambda(
```

```
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS SDK für Kotlin.

`TagResource`

Das folgende Codebeispiel zeigt, wie man es benutzt`TagResource`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- Einzelheiten zur API finden Sie TagResourcein der API-Referenz zum AWS SDK für Kotlin.

`Unsubscribe`

Das folgende Codebeispiel zeigt die Verwendung`Unsubscribe`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }
```

```
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Erstellen einer Amazon-SNS-Anwendung

Das folgende Codebeispiel zeigt, wie eine Anwendung erstellt wird, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

SDK für Kotlin

Zeigt, wie man die Kotlin-API für Amazon SNS verwendet, um eine Anwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer Web-App finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer nativen Android-App finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

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

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
```

```
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für Kotlin.

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic

- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Beispiele für Amazon SQS unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon SQS.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der API-Referenz zum AWS SDK für Kotlin.

Aktionen

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient.fromEnvironment { 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()
    }
}
```

- Einzelheiten zur API finden Sie CreateQueuein der API-Referenz zum AWS SDK für Kotlin.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Einzelheiten zur API finden Sie DeleteMessagein der API-Referenz zum AWS SDK für Kotlin.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Einzelheiten zur API finden Sie DeleteQueuein der API-Referenz zum AWS SDK für Kotlin.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der API-Referenz zum AWS SDK für Kotlin.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- Einzelheiten zur API finden Sie ReceiveMessagein der API-Referenz zum AWS SDK für Kotlin.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
```

```
            messageBody = message
            delaySeconds = 10
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- Einzelheiten zur API finden Sie SendMessagein der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Erstellen einer Nachrichtenanwendung

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS eine Nachrichtenanwendung erstellen können.

SDK für Kotlin

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Beispiele für Step Functions unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Step Functions Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Step Functions

Das folgende Codebeispiel zeigt, wie Sie mit Step Functions beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
        }
    }
```

```
}
```

- Einzelheiten zur API finden Sie ListStateMachinesin der API-Referenz zum AWS SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Aktivität.
- Erstellen Sie einen Zustandsautomaten aus einer Definition von Amazon States Language, die die zuvor erstellte Aktivität als Schritt enthält.
- Führen Sie den Zustandsautomaten aus und reagieren Sie auf die Aktivität mit Benutzereingaben.
- Ermitteln Sie den endgültigen Status und die Ausgabe, nachdem der Lauf abgeschlossen ist, und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
```

```
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.

 You can obtain the JSON file to create a state machine in the following GitHub
 location:

 https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files

 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin code example performs the following tasks:

 1. List activities using a paginator.
 2. List state machines using a paginator.
 3. Creates an activity.
 4. Creates a state machine.
 5. Describes the state machine.
 6. Starts execution of the state machine and interacts with it.
 7. Describes the execution.
 8. Deletes the activity.
 9. Deletes the state machine.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
```

```
    val usage = """
    Usage:
        <roleARN> <activityName> <stateMachineName>

    Where:
        roleName - The name of the IAM role to create for this state machine.
        activityName - The name of an activity to create.
        stateMachineName - The name of the state machine to create.
        jsonFile - The location of the chat_sfn_state_machine.json file. You can
located it in resources/sample_files.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
    val stateMachineName = args[2]
    val jsonFile = args[3]
    val sc = Scanner(System.`in`)
    var action = false

    val polJSON = """{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }"""

    println(DASHES)
    println("Welcome to the AWS Step Functions example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List activities using a Paginator.")
```

```
    listActivitesPagnator()
    println(DASHES)

    println(DASHES)
    println("2. List state machines using a paginator.")
    listStatemachinesPagnator()
    println(DASHES)

    println(DASHES)
    println("3. Create a new activity.")
    val activityArn = createActivity(activityName)
    println("The ARN of the Activity is $activityArn")
    println(DASHES)

    // Get JSON to use for the state machine and place the activityArn value into
it.
    val stream = GetStream()
    val jsonString = stream.getStream(jsonFile)

    // Modify the Resource node.
    val objectMapper = ObjectMapper()
    val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
activityArn)

    // Convert the modified Java object back to a JSON string.
    val stateDefinition = objectMapper.writeValueAsString(root)
    println(stateDefinition)

    println(DASHES)
    println("4. Create a state machine.")
    val roleARN = createIAMRole(roleName, polJSON)
    val stateMachineArn = createMachine(roleARN, stateMachineName, stateDefinition)
    println("The ARN of the state machine is $stateMachineArn")
    println(DASHES)

    println(DASHES)
    println("5. Describe the state machine.")
    describeStateMachine(stateMachineArn)
    println("What should ChatSFN call you?")
    val userName = sc.nextLine()
    println("Hello $userName")
    println(DASHES)
```

```
    println(DASHES)
    // The JSON to pass to the StartExecution call.
    val executionJson = "{ \"name\" : \"$userName\" }"
    println(executionJson)
    println("6. Start execution of the state machine and interact with it.")
    val runArn = startWorkflow(stateMachineArn, executionJson)
    println("The ARN of the state machine execution is $runArn")
    var myList: List<String>
    while (!action) {
        myList = getActivityTask(activityArn)
        println("ChatSFN: " + myList[1])
        println("$userName please specify a value.")
        val myAction = sc.nextLine()
        if (myAction.compareTo("done") == 0) {
            action = true
        }
        println("You have selected $myAction")
        val taskJson = "{ \"action\" : \"$myAction\" }"
        println(taskJson)
        sendTaskSuccess(myList[0], taskJson)
    }
    println(DASHES)

    println(DASHES)
    println("7. Describe the execution.")
    describeExe(runArn)
    println(DASHES)

    println(DASHES)
    println("8. Delete the activity.")
    deleteActivity(activityArn)
    println(DASHES)

    println(DASHES)
    println("9. Delete the state machines.")
    deleteMachine(stateMachineArn)
    println(DASHES)

    println(DASHES)
    println("The AWS Step Functions example scenario is complete.")
    println(DASHES)
}

suspend fun listStatemachinesPagnator() {
```

```
    val machineRequest =
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Aktionen

`CreateActivity`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateActivity`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Einzelheiten zur API finden Sie CreateActivityin der API-Referenz zum AWS SDK für Kotlin.

`CreateStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`CreateStateMachine`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
```

```
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- Einzelheiten zur API finden Sie CreateStateMachinein der API-Referenz zum AWS SDK für Kotlin.

`DeleteActivity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteActivity`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- Einzelheiten zur API finden Sie DeleteActivityin der API-Referenz zum AWS SDK für Kotlin.

`DeleteStateMachine`

Das folgende Codebeispiel zeigt die VerwendungDeleteStateMachine.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteStateMachinein der API-Referenz zum AWS SDK für Kotlin.

`DescribeExecution`

Das folgende Codebeispiel zeigt die VerwendungDescribeExecution.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { 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")
}
```

- Einzelheiten zur API finden Sie DescribeExecutionin der API-Referenz zum AWS SDK für Kotlin.

`DescribeStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStateMachine`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient.fromEnvironment { 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}")
    }
}
```

- Einzelheiten zur API finden Sie DescribeStateMachinein der API-Referenz zum AWS SDK für Kotlin.

`GetActivityTask`

Das folgende Codebeispiel zeigt die Verwendung`GetActivityTask`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

- Einzelheiten zur API finden Sie GetActivityTaskin der API-Referenz zum AWS SDK für Kotlin.

`ListActivities`

Das folgende Codebeispiel zeigt die Verwendung`ListActivities`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { 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}")
        }
```

```
    }
}
```

- Einzelheiten zur API finden Sie ListActivitiesin der API-Referenz zum AWS SDK für Kotlin.

`ListExecutions`

Das folgende Codebeispiel zeigt die Verwendung`ListExecutions`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getExeHistory(exeARN: String?) {
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListExecutionsin der API-Referenz zum AWS SDK für Kotlin.

`ListStateMachines`

Das folgende Codebeispiel zeigt die Verwendung`ListStateMachines`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListStateMachinesin der API-Referenz zum AWS SDK für Kotlin.

`SendTaskSuccess`

Das folgende Codebeispiel zeigt die Verwendung`SendTaskSuccess`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- Einzelheiten zur API finden Sie SendTaskSuccessin der API-Referenz zum AWS SDK für Kotlin.

`StartExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartExecution`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- Einzelheiten zur API finden Sie StartExecutionin der API-Referenz zum AWS SDK für Kotlin.

Support Beispiele für die Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit verwenden. Support

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Support

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von Support beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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.fromEnvironment { 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++
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeServicesin der API-Referenz zum AWS SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie verfügbare Services und Schweregrade für Fälle ab und zeigen Sie sie an.
- Erstellen Sie einen Supportfall mit einem ausgewählten Service, einer ausgewählten Kategorie und einem ausgewählten Schweregrad.
- Rufen Sie eine Liste der offenen Fälle für den aktuellen Tag ab und zeigen Sie sie an.
- Fügen Sie dem neuen Fall einen Anhangssatz und eine Mitteilung hinzu.
- Beschreiben Sie den neuen Anhang und die Mitteilung für den Fall.
- Lösen Sie den Fall.
- Rufen Sie eine Liste der gelösten Fälle für den aktuellen Tag ab und zeigen Sie sie an.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Aktionen

`AddAttachmentsToSet`

Das folgende Codebeispiel zeigt, wie man es benutzt`AddAttachmentsToSet`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- Einzelheiten zur API finden Sie AddAttachmentsToSetin der API-Referenz zum AWS SDK für Kotlin.

`AddCommunicationToCase`

Das folgende Codebeispiel zeigt die VerwendungAddCommunicationToCase.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }
```

```
    SupportClient.fromEnvironment { 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")
        }
    }
}
```

- Einzelheiten zur API finden Sie AddCommunicationToCasein der API-Referenz zum AWS SDK für Kotlin.

`CreateCase`

Das folgende Codebeispiel zeigt die Verwendung`CreateCase`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- Einzelheiten zur API finden Sie CreateCasein der API-Referenz zum AWS SDK für Kotlin.

`DescribeAttachment`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAttachment`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- Einzelheiten zur API finden Sie DescribeAttachmentin der API-Referenz zum AWS SDK für Kotlin.

`DescribeCases`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCases`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.fromEnvironment { 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}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeCasesin der API-Referenz zum AWS SDK für Kotlin.

`DescribeCommunications`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCommunications`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { 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 ""
}
```

- Einzelheiten zur API finden Sie DescribeCommunicationsin der API-Referenz zum AWS SDK für Kotlin.

`DescribeServices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeServices`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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.fromEnvironment { 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
}
```

- Einzelheiten zur API finden Sie DescribeServicesin der API-Referenz zum AWS SDK für Kotlin.

`DescribeSeverityLevels`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSeverityLevels`.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { 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
    }
}
```

- Einzelheiten zur API finden Sie DescribeSeverityLevelsin der API-Referenz zum AWS SDK für Kotlin.

`ResolveCase`

Das folgende Codebeispiel zeigt die VerwendungResolveCase.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- Einzelheiten zur API finden Sie ResolveCasein der API-Referenz zum AWS SDK für Kotlin.

Beispiele für Amazon Translate unter Verwendung von SDKs für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das AWS SDK für Kotlin mit Amazon Translate verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Amazon-SNS-Anwendung

Das folgende Codebeispiel zeigt, wie eine Anwendung erstellt wird, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

SDK für Kotlin

Zeigt, wie man die Kotlin-API für Amazon SNS verwendet, um eine Anwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer Web-App finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer nativen Android-App finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Sicherheit für die AWS SDK für Kotlin

Cloud-Sicherheit genießt bei Amazon Web Services (AWS) höchste Priorität. Als AWS -Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die zur Erfüllung der Anforderungen von Organisationen entwickelt wurden, für die Sicherheit eine kritische Bedeutung hat. Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Im Modell der übergreifenden Verantwortlichkeit wird Folgendes mit „Sicherheit der Cloud“ bzw. „Sicherheit in der Cloud“ umschrieben:

Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der alle in der AWS Cloud angebotenen Dienste ausgeführt werden, und für die Bereitstellung von Diensten, die Sie sicher nutzen können. Unsere Sicherheitsverantwortung hat bei uns höchste Priorität AWS, und die Wirksamkeit unserer Sicherheit wird im Rahmen der AWS Compliance-Programme regelmäßig von externen Prüfern getestet und verifiziert.

Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem von Ihnen genutzten AWS Dienst und anderen Faktoren, wie der Sensibilität Ihrer Daten, den Anforderungen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Themen

- Datenschutz in AWS SDK für Kotlin
- AWS SDK für Kotlin Unterstützung für TLS 1.2
- Identitäts- und Zugriffsverwaltung
- Überprüfung der Einhaltung der Vorschriften für dieses AWS Produkt oder diese Dienstleistung
- Ausfallsicherheit für dieses AWS Produkt oder diese Dienstleistung
- Sicherheit der Infrastruktur für dieses AWS Produkt oder diesen Service

Datenschutz in AWS SDK für Kotlin

Das Modell der AWS gemeinsamen Verantwortung und gilt für den Datenschutz in AWS SDK für Kotlin. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit dem SDK für Kotlin oder anderen Geräten AWS-Services über die Konsole, API oder arbeiten. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL

für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

AWS SDK für Kotlin Unterstützung für TLS 1.2

Die folgenden Informationen gelten nur für die Java-SSL-Implementierung (die Standard-SSL-Implementierung in der AWS SDK für Kotlin Ziel-JVM). Wenn Sie eine andere SSL-Implementierung verwenden, erfahren Sie in Ihrer spezifischen SSL-Implementierung, wie Sie TLS-Versionen erzwingen.

TLS-Unterstützung in Java

TLS 1.2 wird ab Java 7 unterstützt.

Vorgehensweise zum Überprüfen der TLS-Version

Um zu überprüfen, welche TLS-Version in Ihrer Java Virtual Machine (JVM) unterstützt wird, können Sie den folgenden Code verwenden.

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.joinToString(separator
 = ", "))
```

Um den SSL-Handshake in Aktion zu sehen und welche Version von TLS verwendet wird, können Sie die Systemeigenschaft javax.net.debug verwenden.

```
-Djavax.net.debug=ssl
```

Identitäts- und Zugriffsverwaltung

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. AWS IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten

- Verwalten des Zugriffs mit Richtlinien
- Wie AWS-Services arbeiten Sie mit IAM
- Fehlerbehebung bei AWS Identität und Zugriff

Zielgruppe

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, in der Sie tätig sind. AWS

Dienstbenutzer — Wenn Sie dies AWS-Services für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr AWS Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Falls Sie auf eine Funktion nicht zugreifen können AWS, finden Fehlerbehebung bei AWS Identität und Zugriff Sie weitere Informationen in der Bedienungsanleitung der von AWS-Service Ihnen verwendeten.

Serviceadministrator — Wenn Sie in Ihrem Unternehmen für die AWS Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf AWS. Es ist Ihre Aufgabe, zu bestimmen, auf welche AWS Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM verwenden kann AWS, finden Sie in der Benutzeranleitung des von AWS-Service Ihnen verwendeten.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf AWS verfassen können. Beispiele für AWS identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie im Benutzerhandbuch der AWS-Service von Ihnen verwendeten.

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung

oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind)

sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die sich daraus ergebenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

Wie AWS-Services arbeiten Sie mit IAM

Einen allgemeinen Überblick darüber, wie die meisten IAM-Funktionen AWS-Services funktionieren, finden Sie im AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren.

Informationen zur Verwendung bestimmter Dienste AWS-Service mit IAM finden Sie im Abschnitt Sicherheit im Benutzerhandbuch des jeweiligen Dienstes.

Fehlerbehebung bei AWS Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit AWS und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `awes:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `awes:GetWidget`-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an AWSübergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten

(ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob diese Funktionen AWS unterstützt werden, finden Sie unter. Wie AWS-Services arbeiten Sie mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Überprüfung der Einhaltung der Vorschriften für dieses AWS Produkt oder diese Dienstleistung

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen

zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Ausfallsicherheit für dieses AWS Produkt oder diese Dienstleistung

Die AWS globale Infrastruktur basiert auf AWS-Regionen Availability Zones.

AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind.

Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Sicherheit der Infrastruktur für dieses AWS Produkt oder diesen Service

Dieses AWS Produkt oder dieser Dienst verwendet Managed Services und ist daher durch die AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf dieses AWS Produkt oder diesen Service zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS -Security-Token-Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Dokumentverlauf

In diesem Thema werden wichtige Änderungen beschrieben, die im Laufe der Geschichte des AWS SDK für Kotlin Entwicklerhandbuchs vorgenommen wurden.

Änderung	Beschreibung	Datum
the section called “Erneute Versuche”	Der Inhalt wurde neu organisiert und es wurden Details zu Ausnahmen hinzugefügt, die wiederholt werden können.	10. September 2025
Fügen Sie Informationen dazu hinzu, wie Anmeldeinformationen für einen eigenständigen Anbieter von Anmeldeinformationen zwischengespeichert werden	Anmeldeinformationen bei einem eigenständigen Anbieter zwischenspeichern	17. Juni 2025
the section called “Spöttisch”	Fügen Sie Informationen über Mocking und die Verwendung von MockK mit dem SDK für Kotlin hinzu	30. April 2025
Fügen Sie Informationen zur Lösung von Abhängigkeitskonflikten mit dem SDK mithilfe von Gradle hinzu	Wie löse ich Abhängigkeitskonflikte?	15. April 2025
Schutz der Datenintegrität mit Prüfsummen	Der Inhalt wurde mit Einzelheiten zur automatischen Prüfsummenberechnung aktualisiert.	16. Januar 2025
Inhalt der Anbieterkette mit Standardanmeldedaten aktualisieren	Die Anbieterkette für Standardanmeldedaten.	15. Januar 2025

Beispiele für Build-Dateien für Updates	Zeigt Build-Dateielemente an, wie sie von Gradle Version 8.11.1 generiert wurden. Zeige die Verwendung von BOM. Links zur neuesten Version von Artefakten einbetten.	18. Dezember 2024
Thema DynamoDB Mapper hinzufügen (Developer Preview)	Zuordnen von Klassen zu DynamoDB-Elementen mithilfe des DynamoDB-Mapper (Developer Preview)	29. Oktober 2024
Amazon S3 S3-Bucket-Namen aktualisieren	Amazon S3 S3-Prüfsummen mit AWS SDK für Kotlin	30. September 2024
Fügen Sie der OkHttp 4-Engine Informationen hinzu	Geben Sie einen HTTP-Engine-Typ an	26. September 2024
Informationen zu AWS kontobasierten Endpunkten für DynamoDB hinzufügen	AWS Verwenden Sie kontobasierte Endpunkte	24. September 2024
Fügen Sie ein Thema zur Fehlerbehebung hinzu FAQs	Fehlersuche FAQs	18. September 2024
Beispiel für die OpenTelemetry Update-Konfiguration und die Konfiguration des globalen Standardtelemetrie-Anbieters	Beobachtbarkeit	2. Mai 2024
Geben Sie weitere Informationen zum Prozess zur Erstellung von Serviceclienten an	Erstellen Sie einen Serviceclient	14. März 2024
Thema Access Point für mehrere Regionen hinzufügen	Arbeiten Sie mit Amazon S3 Multiregion Access Points mithilfe des SDK für Kotlin	6. Februar 2024

Fügen Sie Anweisungen zum Katalog der Gradle-Version hinzu	Gradle-Versionskatalog (Registerkarte)	19. Dezember 2023
Version zur allgemeinen Verfügbarkeit	AWS SDK für Kotlin Entwickle rhandbuch	27. November 2023
Aktualisieren Sie den Konfigurationsabschnitt für die Client-Endgeräte auf der Grundlage von SDK-Updates	Client-Endpunkte	25. August 2023
Amazon S3 S3-Prüfsummen	Abschnitt zur Verwendun g flexibler Prüfsummen mit Amazon S3 hinzugefügt.	14. August 2023
Thema Observability hinzufügen	Beobachtbarkeit	3. August 2023
Fügen Sie ein Thema hinzu, das Wiederholungsversuche behandelt	Erneute Versuche	07. Juli 2023
Aktualisieren Sie den Abschnitt zur HTTP-Clie nt-Konfiguration auf der Grundlage von SDK-Updates	HTTP-Client-Konfiguration	6. Juni 2023
Thema HTTP-Vorsignierung hinzufügen	Vorsignieren von Anfragen	02. Juni 2023
Thema HTTP-Interzeptoren hinzufügen	HTTP-Interzeptoren	22. Mai 2023
Support für automatische Token-Aktualisierung	Aktualisieren Sie die Anweisungen für den Single Sign-On-Zugriff.	18. Mai 2023

Amazon S3 S3-Prüfsummen	Fügen Sie einen Abschnitt hinzu, der beschreibt, wie Prüfsummen mit Amazon S3 verwendet werden.	15. Mai 2023
Überschreiben Sie die Service-Client-Konfiguration	Fügen Sie einen Abschnitt hinzu, in dem beschrieben wird, wie die Konfiguration eines Service-Clients außer Kraft gesetzt wird, und in dem beschrieben wird, wie Ressourcen davon betroffen sind.	8. Mai 2023
Erzwingen Sie eine TLS-Mindestversion	Fügen Sie einen Abschnitt hinzu, in dem die Optionen zur Durchsetzung einer TLS-Mindestversion beschrieben werden.	3. Mai 2023
Konfiguration des Client-Endpunkts	Fügen Sie ein Thema hinzu, das die Konfiguration von Client-Endpunkten behandelt.	7. April 2023
Aktualisierungen der bewährten Methoden für IAM	Aktualisierung des Leitfadens zur Ausrichtung an bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte Sicherheitsmethoden in IAM.	22. März 2023
Fügen Sie eine Beispieldatei für ein Maven-Projekt hinzu	Zeigen Sie im Thema Einrichten ein Beispiel für eine Maven-Projektdatei zusätzlich zu einer Projektdatei in Gradle.	2. Dezember 2022

Überarbeiten Sie den Inhalt der Vorschau des Entwickle rhandbuchs	Der Inhalt wurde aktualisiert, um aktuelle Entwicklungsarbeit en widerzuspiegeln	4. Oktober 2022
AWS SDK für Kotlin Vorschauversion für Entwickler	AWS SDK für Kotlin	2. Dezember 2021
AWS SDK für Kotlin Alpha-Ver sion	Ankündigung einer neuen AWS SDK für Kotlin Alpha-Ver sion	30. August 2021

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.